UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria de Manresa

Treball Final de Grau

Integració d'un braç robot en un magatzem intel·ligent

Autor/a: Adrià Arias Morales

Director/a: Teresa Escobet Canal

Grau en Enginyeria Electrònica Industrial i Automàtica

Curs 2017-2018

RESUM DEL PROJECTE

L’ús de robots és cada vegada més gran i es troben presents en àmbits industrials i de serveis. En aquest projecte, es presenta la implementació d’un braç robotitzat en una estació de treball, el qual mitjançant una càmera de visió artificial, és capaç de poder classificar peces en funció del color i de la geometria que presenti, en concret, quadrat, rectangular i triangular.

En aquest treball, s’ha volgut representar el concepte de procés industrial de classificat en magatzem intel·ligent mitjançant un braç robot prefabricat de la marca Tinkerkit, una càmera de visió artificial de baixes prestacions i coordinat en mesura de lo possible amb software i hardware de caràcter lliure.

Pel que fa el software, s’ha fet ús del Unity 3D, per a la planificació de trajectòries amb simulació d’animacions 3D. Finalment, per el control dels motors del braç robot i la interacció amb la Pixy CMUcam5, s’ha fet servir una placa de control anomenada Arduino Uno.

ABSTRACT

The use of robots is getting bigger and they are present in industrial and service environments. In this project, we present the implementation of a robotized arm in a workstation, which by means of an artificial vision camera, is able to classify pieces according to the color and the geometry presented, in particular, square, rectangular and triangular.

In this work, we wanted to represent the concept of industrial process of classified in intelligent warehouse by means of a prefabricated robot arm of the Tinkerkit brand, a low-performance artificial vision camera and coordinated as far as possible with software and hardware of free character.

As for the software, Unity 3D has been used to plan trajectories with simulation of 3D animations. Finally, for the control of the robot arm motors and the interaction with the Pixy CMUcam5, a control plate called Arduino Uno has been used.

Índex

Índex de Figures

Índex de Taules

1. INTRODUCCIÓ

1.1. Context del treball

És una realitat que estem davant l'era de l'emmagatzematge intel·ligent (t21mx, 2018). A causa de les necessitats del mercat, les companyies estan construint grans superfícies com a punts de distribució, dotant-los de l'equip necessari per a disposar de la seva mercaderia de la manera més adequada. Capacitat, distància dels punts de comercialització i disposició de productes són alguns dels elements clau a l'hora d'especificar la ubicació, construcció i organització d'un punt de distribució. Tot ha d'estar relacionat favorablement per a una operació adequada del negoci.

Avui dia, el concepte d’emmagatzemar va més enllà de l'acció que suposa guardar alguna cosa, ja que s'ha convertit en una activitat que requereix la coordinació de diferents processos tecnològics que contribueixen directament al creixement d'una companyia. Utilitzar processos automatitzats dins d'un magatzem, com per exemple, l’ús de robots, per tasques d’etiquetatge, manipulació, visió y organització de productes, ha contribuït a generar beneficis gairebé de manera immediata, millorant en aspectes com ara una major velocitat en les activitats del punt de distribució, un major control de mercaderies, a més d'una millora en la precisió de resultats i una reducció de la mà d'obra.

És degut això, que la gran majoria de processos d’automatització industrials ja es troben robotitzats. Caldria potenciar la recerca i implantació de sensors i actuadors que fessin encara més eficient la tasca d’aquests elements.

D’altre banda, som partíceps del gran desenvolupament de tecnologies en l’àmbit de la intel·ligència artificial (IA), és el cas de reconeixements tan de colors com de formes mitjançant càmeres i sensors òptics.

En definitiva, la implementació del conjunt d’aquests elements, es pot veure reflectit tant en la vida diària, com en millores de processos industrials.

1.2. Objectius

L'objectiu principal del treball és la implementació d'un braç robotitzat en una estació de treball amb la capacitat d'identificar, seleccionar i classificar certs objectes mitjançant la incorporació d'elements d'intel·ligència artificial, capaç de comunicar-se amb un controlador per gestionar i temporitzar el procés.

La versatilitat d'aquest sistema ha de permetre la seva aplicació en diferents tasques o seccions de producció i emmagatzematge. És per això, que el hardware fet servir permet una amplia compatibilitat.

1.3. Motivacions

Les motivacions inicials que m'han portat a realitzar aquest projecte són fonamentalment:

- L'aprofitament de certes plataformes de hardware que es troba en disposició i/o tipificació de lliure accés, modificació i implementació.
- Programació i variació de codis de microcontroladors.
- Utilització de programes de realitat virtual per a simulacions.
- Posada en pràctica dels coneixements adquirits durant el període de formació.

De les motivacions esmentades anteriorment, he trobat força interessant el potencial que tenen el conjunt del hardware de caràcter lliure, amb la seva programació de codi.

Altrament, m'ha aportat coneixements el treballar amb elements mecatrònics.

1.4. Descripció del funcionament

El sistema plantejat consta d'un braç robotitzat gestionat amb un microcontrolador amb la incorporació d'un sistema de visió artificial per el desenvolupament d'una tasca d'identificació, selecció i classificació en una estació de treball.

La càmera estarà connectada a un microcontrolador en el qual resideix un programa que te per objectiu reconèixer el tipus de peça, en concret forma i color. En funció del resultat del reconeixement de la peça, el braç robot la situarà en un lloc d'un dels dos magatzems predefinits.

1.5. Aplicacions

L’àmbit d’ús d’aquest sistema es pot adreçar a diferents aplicacions;

- Principalment a la gestió d’un magatzem intel·ligent.
- Processos industrials repetitius d’alta exigència i precisió.
- L’ús d’element d’ IA en àmbits de presa de decisions.
- Aplicació en processos quirúrgics.

2. ANTECEDENTS

El muntatge del braç robot i del magatzem, s'integrarà a la estació de treball que es mostra a la Figura 1.

Figura 1. Vista general estació 4.

La última de les etapes del procés és l'emmagatzematge de les peces fabricades en un lloc determinat pel codi que porta el palet. En aquest cas, el manipulador de l'estació ha d'extreure la caixa del porta palet i portar-la a una zona determinada del magatzem. La zona de descàrrega queda determinada per la codificació del palet.

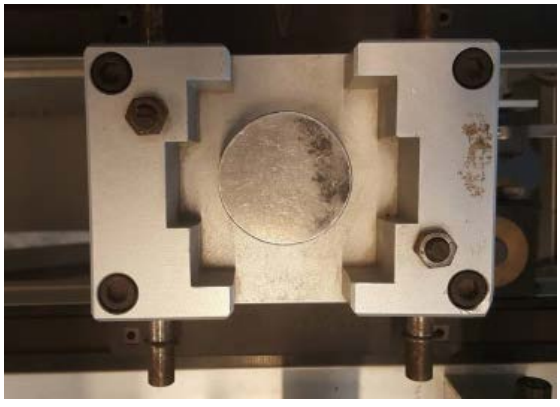


Figura 2. Palet.

Aquest manipulador consta del pistó (F) que suporta una pinça de subjecció i permet desplaçar la caixa verticalment. El moviment horitzontal, al igual que l'estació 3, es realitza mitjançat un vis sens fi activat per un motor. Degut a que el desplaçament del manipulador és lineal i continu, hi ha dos finals de carrera de seguretat, a cada extrem del recorregut, que paren el motor.

També disposem del pistó A que és l'encarregat de desplaçar el palet de sortida de les peces. El transport del magatzem al palet es realitzarà mitjançant un robot (no disponible en aquests moments).

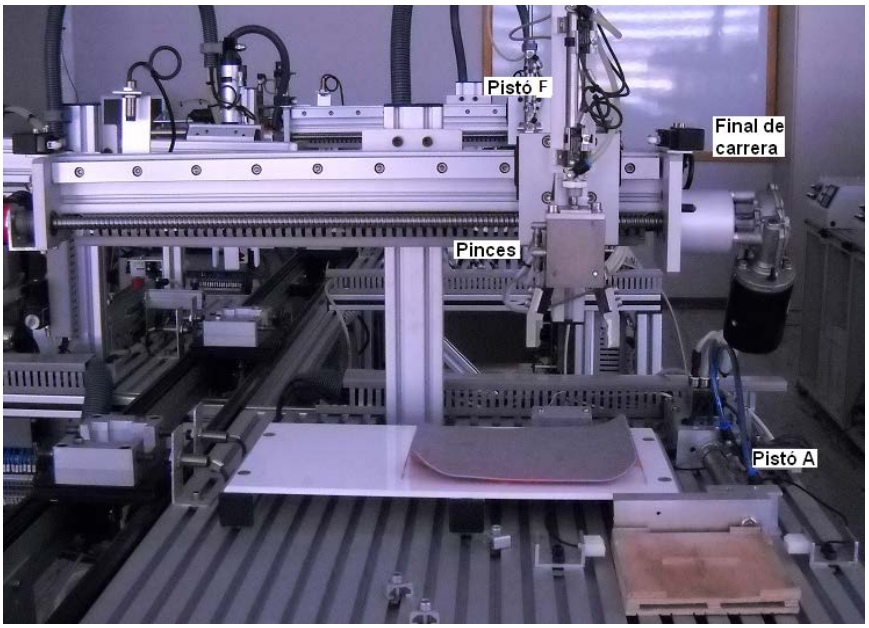


Figura 3. Identificació dels pistons de l'estació 4.

Aquesta estació de treball està comandada per un autòmat programable industrial (API) de la marca OMRON model CP1L. Aquest API està connectat a Ethernet i es pot observar a la següent figura:

Figura 4. PLC OMRON CP1L.

3. HARDWARE UTILIZAT

3.1. Arduino

Arduino (anteriorment conegut com Genuino a nivell internacional fins a octubre de 2016), és una companyia de codi obert i de maquinari obert, així com un projecte i comunitat internacional que dissenya i fabrica plaques de desenvolupament de maquinari per a construir dispositius digitals i dispositius interactius que puguin censar i controlar objectes del món real (Arduino, 2018). Arduino es centra a apropar i facilitar l'ús de l'electrònica i la programació de sistemes en projectes multidisciplinaris. Els productes que venen a la companyia estan distribuïts com a Maquinari i Programari Lliure, sota la Llicència Pública General Reduïda de GNU (LGPL) o la Llicència Pública General de GNU (GPL), permetent la fabricació de les plaques Arduino i distribució del programari per qualsevol individu.

De totes les plaques produïdes per Arduino, s’ha escollit Arduino Uno per fer el treball ja que és la millor per iniciar-se amb l’electrònica i codificació.

Arduino Uno és una placa de microcontrolador basada en ATmega328P (Figura 5). Té 14 pins digitals d'entrada / sortida (dels quals 6 es poden utilitzar com a sortides PWM), 6 entrades analògiques, una cristall de quars de 16 MHz, un connector USB, un connector jack, un encapçalat ICSP i un botó de RESET. Conté tot el necessari per suportar el microcontrolador; simplement es connecta a un PC amb un cable USB o amb un adaptador de CA a CC o una bateria per començar.

Es pot fer servir l’Arduino Uno sense preocupar-se massa de fer alguna cosa incorrecte, ja que en el pitjor dels casos, es pot reemplaçar el xip. "Uno" significa un en italià i va ser elegit per marcar el llançament del programari Arduino (IDE) 1.0. La placa Uno i la versió 1.0 d'Arduino Software (IDE) van ser les versions de referència d'Arduino, ara evolucionades a versions més recents. Arduino Uno és el primer d'una sèrie de plaques USB Arduino, i el model de referència per a la plataforma Arduino.

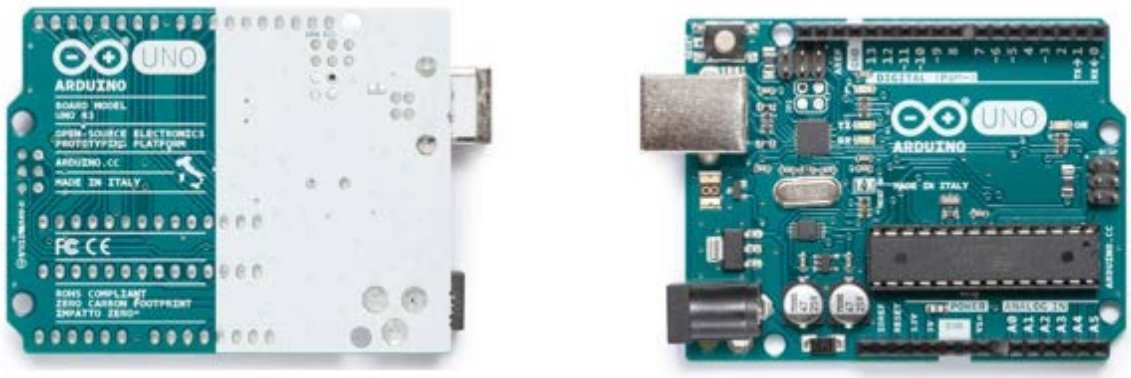


Figura 5. Frontal i posterior placa Arduino.

A continuació es mostra una taula amb les característiques tècniques;

Taula 1. Característiques tècniques Arduino Uno.

Microcontrolador	ATmega328P
Voltatge de funcionament	5V
Voltatge d'entrada (recomanat)	7-12V
Voltatge d'entrada (límit)	6-20V
Pines d'E / S digitals	14 (dels quals 6 proporcionen sortida PWM)
Pins PWM Digital E/S	6
Pins d'entrada analògics	6
Corrent DC per pins de E/S	20 mA
Corrent DC per 3.3V	50 mA
Memòria Flash	32 KB (ATmega328P) dels quals 0,5 KB usats pel gestor d'arrencada
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Velocitat del rellotge	16 MHz
LEDS	13
Longitud	68.6 mm
Amplada	53.4 mm
Pes	25 g

3.2. Braç Robot

Degut a que la placa de control que s'utilitza, és l'Arduino Genuino/Uno, s'ha decidit emprar un Braz Robot preconfigurat i de la mateixa marca Tinkerkit Braccio Arduino, (RS-Online, 2018).

Figura 6. Braç robot Tinkerkit.

El braç robot Tinkerkit està dissenyat per fer-ne un ús "d'escriptori", i s'adquireix en forma de kit per tal de muntar-ho i configurar-ho segons s'ajusti a les nostres necessitats. És un braç fàcilment adaptable i per això es pot utilitzar per varies tasques.

Pot recollir i moure objectes, muntar una càmera y que segueixi els moviments mitjançant una videotrucada o instal·lar un panel solar i que segueixi el moviment del sol.

A continuació s'adjunta la taula de característiques:

Taula 2. Característiques Braccio Tinkerkit.

Abast horitzontal:	80 cm
Abast vertical:	52 cm
Amplada de la base:	14 cm
Amplada de la pinça:	9 cm
Longitud del cable:	40 cm
Màxima capacitat de carrega a una distancia de 32 cm	150 g
Pes màxim de la base:	400 g
Servomotors:	2 x SR 311, 4 x SR431
Peso total:	0.792 kg
Voltatge de funcionament:	5 V
Consumo:	20 mW
Màxima corrent:	1.1 A

3.2.1. Disseny mecànic

L'aspecte general del disseny del braç esta pensat per complir les nostres necessitat, ja que s'adapta fàcilment a l'entorn de treball. Tot i tindre unes dimensions reduïdes, les característiques generals detallades anteriorment, ens fan veure que és una bona opció a l'hora de desenvolupar les nostres idees.

L'estructura d'aquest braç, esta formada per 6 parts (pinça, canell, braç inferior, braç superior i base), les quals són imprescindibles per dur a terme qualsevol tasca, amb una mínima exigència:

- **Pinça**

La pinça utilitzada es mostra a la Figura 7. És una de les parts més importants dins de l'estructura, degut que ens proporciona una obertura de fins a 9 cm. Cal destacar el disseny, ja que gràcies a la seva forma allargada, ens permetrà introduir-la en espais reduïts per tal d'agafar qualsevol objecte.

Els moviments d'obertura i de tancat ens el proporcionarà un servomotor conjuntament amb un petit engranatge ja inclòs al mateix disseny de la pinça. El servomotor, es trobarà adjunt a un extrem de la pinça on també s'hi troba situat un dels dos engranatges simètrics, per tal de facilitar el moviment del conjunt d'aquesta.

L'acoblament de cada una de les parts de la pinça es farà amb cargols, femelles i volanderes.
Degut a l'amplitud màxima de 9 cm que pot adoptar la pinça, conjuntament amb la resta de parts del prototip, ens permetran treballar amb certa comoditat.

Figura 7. Pinça.

- **Canell**

Es pot considerar una de les parts més interessants del braç robot, ja que té la funció de moure la pinça verticalment i amb un moviment rotacional. Aquest component es mostra a la (Figura 8).

El funcionament del canell, està governat per un servomotor sostingut per 4 cargols a l'extrem superior i el qual s'acoblarà a la base de la pinça, per tal de fer-la moure.

Figura 8. Canell.

- **Braç Inferior o Colze**

Dins del disseny del robot, és una articulació amb una funció semblant a la del canell, ja que ens permetrà realitzar moviments amb una amplitud de 180º.

Aquesta part de l'estructura, es divideix en dues peces simètriques que s'acoblaran entre elles situant un servomotor a l'extrem superior d'aquestes. El servo, es sostindrà amb 4 cargols a una de les parts simètriques del braç inferior, prèviament a acoblar-se amb l'altre meitat, que també ho farà mitjançant 4 cargols.

Figura 9. Meitat peça braç inferior/colze.

- **Braç superior**

Realitza la mateixa funció que el braç inferior/colze, ja que està format per dues peces simètriques que s'uneixen entre elles situant un servomotor (M3) a l'extrem superior d'aquest, permetent realitzar moviments amb una amplitud de 180º. Aquesta part superior, s'acoblarà amb la part inferior del braç inferior/colze.

El servo es sostindrà amb 4 cargols per la part interior de una de les dues peces simètriques, prèviament a unir-se amb l'altre meitat, que també ho farà mitjançant 4 cargols.

Figura 10. Meitat peça braç superior.

- **Base**

Aquesta part del conjunt de l'estructura, podem dividir-la en dos components ben diferenciats: la base pròpiament dita, i la fixació del braç superior amb la base.

La primera, és una petita plataforma que subjectarà el braç robot. Està governada per un servomotor (M1), que s'introduirà al seu interior acoblada amb 4 cargols per la part exterior. La base és l'encarregada de realitzar el moviment rotacional del braç robot, amb una amplitud de fins 180º.

Figura 11. Base braç robot.

En el cas de l'altre component que forma la base, realitza la funció de fixar l'extrem inferior del braç superior amb la base, mitjançant dues peces que aniran enclavades a banda i banda.
A més de fixar les dues parts del braç robot, estarà governat per un servomotor (M2) que facilitarà el moviment del braç superior.

Figura 12. Peça per fixar base braç robot.

3.2.2. Actuadors

L'estructura del braç robot Tinkerkit, la conformen el conjunt de peces que funcionaran com articulacions, i els servo motors que ho faran com actuadors, ja que permetran realitzar el moviment d'aquestes amb una certa amplitud.

El conjunt del braç estarà governat per 6 servo motor (4 servo motors SR 431 i 2 servo motors SR311). La distribució d'aquest serà la següent:

- Motor 1 (SR 431) -> Moviment Base : Aquest servomotor connecta amb el pin 4 de la Braccio Shield V.4.
- Motor 2 (SR 431) -> Moviment Braç superior/Hombro : Aquest servomotor connecta amb el pin 2 de la Braccio Shield V.4.
- Motor 3 (SR 431) -> Moviment Braç inferior/Colze: Aquest servomotor connecta amb el pin 9 de la Braccio Shield V.4.
- Motor 4 (SR 431) -> Moviment vertical Canell: Aquest servomotor connecta amb el pin 6 de la Braccio Shield V.4.
- Motor 5 (SR 311) -> Moviment rotacional Canell: Aquest servomotor connecta amb el pin5 de la Braccio Shield V.4.
- Motor 6 (SR 311) -> Moviment Pinces: Aquest servomotor connecta amb el pin 3 de la Braccio Shield V.4.

Figura 14. Els 4 servo motors SR 431.

Figura 13. Servo motor SR 311.

3.2.3. Braccio Shield V.4

Braccio Shield V.4 és una Shield prefabricada que es pot adquirir comprant el braç robot Braccio Tinkerkit que es fa servir en aquest projecte.

Conceptualment, les Shields són plaques de circuits modulars que es munten unes sobre d'altres per donar funcionalitat extra a un Arduino. En el nostre cas, la Braccio Shield V.4, ens serà imprescindible per poder governar fins a 6 servo motors, com els mostrats a la Figura 13 i Figura 14 i els quals, s'encarregaran de moure les diferents articulacions del braç robot.

També disposa de 6 ranures (inputs), que ens proporcionen la possibilitat d'afegir components com sensors.

Pel que fan la transmissió de dades, la Braccio Shield V.4 presenta un connector TWI, el qual també es coneix com la comunicació I2C. Aquest protocol de comunicació fa servir el SDA (dades), SCL (senyal de rellotge) i GND (terra).

Addicionalment , trobem un connector Serial, també conegut com UART i que fa servir dues línies de comunicació de dades, RX (recepció) i Tx (transmissió).

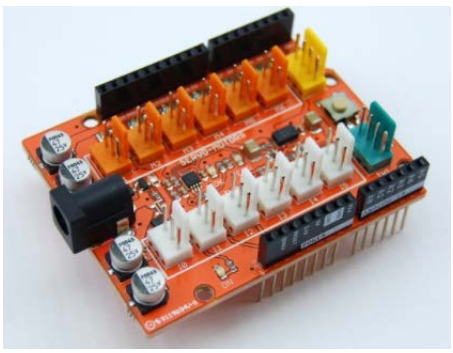


Figura 15. Braccio Shield V.4.

3.3. Càmera de visió

3.3.1. Pixy CMUcam5

La càmera que s'ha implementat conjuntament amb el braç robot ha estat el Pixy CMUCam5 (BricoGeek, 2018), i es pot veure a la (Figura 16). S'ha escollit aquesta càmera ja que és compatible amb Arduino i disposa de força suport dins de la comunitat. El Pixy CMUCam 5 és un sensor d'imatge amb un potent processador que es pot programar per enviar només la informació que està buscant, exporta informació per port serial UART, SPI, I2C, digital out, o analògiques. També es pot utilitzar per identificar to i saturació d'imatges. Això vol dir que la il·luminació o l'exposició no afectaran la detecció de la Pixy CMUCam.

Posseeix una capacitat de processament a 50 fotogrames per segon i es pot configurar perquè només enviï imatges que s'ha dit específicament que s'han de buscar. Té una aplicació de codi obert anomenat PixyMon per al seu control i prova que es pot descarregar gratuïtament des del seu web.

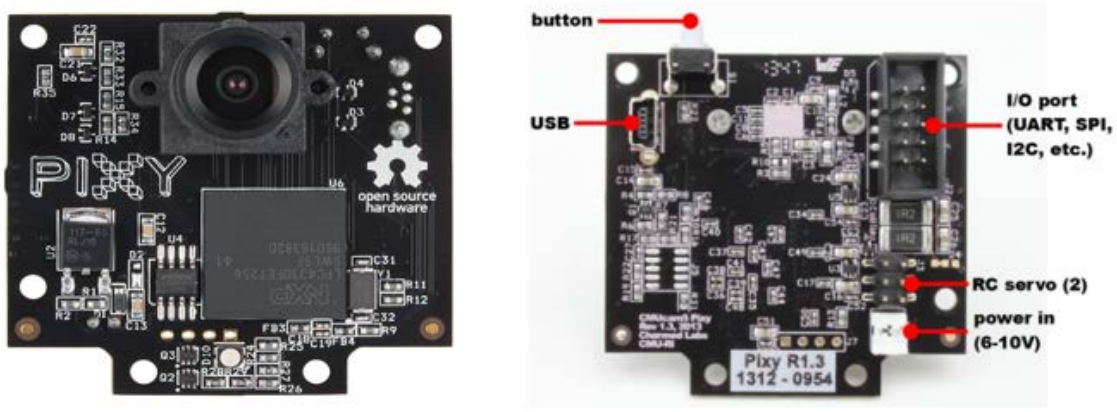


Figura 16. Frontal i posterior de la càmera Pixy.

A continuació s'adjunta la taula de característiques que la conformen:

Tabla 1. Característiques PixyCMUcam5.

Processador	NXP LPC4330, 204 MHz, dual core
Memòria Ram	264Kb
Consum	140mA
Sensor d'Imatge	Omnivision OV9715, 1/4", 1280X800
Angle de visió	75 graus horitzontals, 47 graus verticals
Reconeixement d'imatge	Pixy reconeix l'objecte premen el botó
Simplificació de programació	Rep només els objectes que vols detectar
Ports de comunicació	SPI, I2C, UART, USB o sortida analógica/digital.
Dimensions (sense cable IDC i sense cargols)	50mm x 54 mm x 2mm
Alçada de la càmera	25 mm
Pes (sense cable IDC i sense cargols)	25,5 g

S'entrega cada Pixy CMUcam5 amb un cable de 6 pins a 10 pins IDC i cargols de muntatge.

3.4. Comunicació entre tots els elements

3.4.1. Comunicació entre Arduino Uno i Braccio Tinkerkit

Per realitzar la comunicació entre l'Arduino i el braç robot, primerament, s'acobla la Shield a sobre de l'Arduino Uno, fen coincidir les potes dels pins analògics i digitals de la Shield, amb les de l'Arduino Uno.

Els servomotors, s'han connectat a la Braccio Shield als respectius pins d'entrada dels servo motors situats a la part superior i ja enumerats, com podem observar a la següent figura.

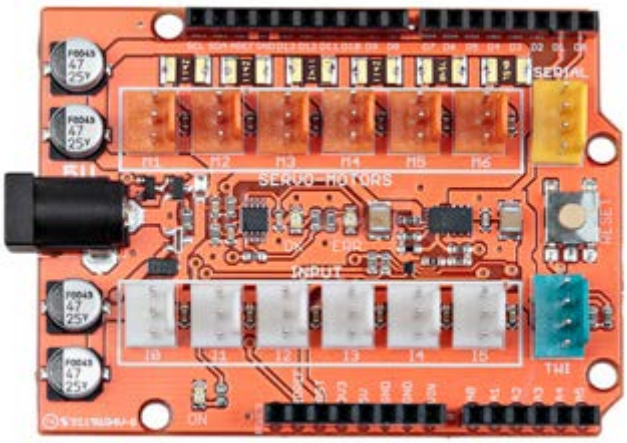


Figura 17. Vista superior Braccio Shield V.4.

Per comprovar que la comunicació entre Arduino Uno i el braç robot és la correcte, s'ha carregat un petit codi (TestBraccio90) que ens servirà per alinear els motors de cada articulació (Anex 1).

3.4.2. Comunicació entre Arduino Uno i Pixy CMUcam5

Pixy disposa d'un seguit de ports de comunicació per tal de transmetre dades, com per exemple, SPI, I2C, UART o USB. El bus de comunicació emprat en aquest projecte per transmetre dades entre Arduino i el Pixy CMUcam5 és el SPI.

3.4.3. Bus SPI

El bus SPI (Serial Peripheral Interface) va ser desenvolupat per Motorola en 1980 (Llamas, 2018). Els seus avantatges respecte a altres sistemes han fet que es converteixi en un estàndard de facto en el món de l'electrònica i automatització.

El bus SPI té una arquitectura de tipus mestre-esclau (Figura 18). El dispositiu mestre (màster) pot iniciar la comunicació amb un o diversos dispositius esclaus (slave), i enviar o rebre dades d'ells. Els dispositius esclaus no poden iniciar la comunicació, ni intercanviar dades entre ells directament.

La comunicació de dades entre mestres i esclau es realitza en dues línies independents, una del mestre als esclaus, i una altra dels esclaus al mestre. Per tant la comunicació és Full Duplex, és a dir, el mestre pot enviar i rebre dades simultàniament.

Una altra característica del SPI és que es tracta d’un bus síncron. El dispositiu mestre proporciona un senyal de rellotge, que manté a tots els dispositius sincronitzats.

Això redueix la complexitat del sistema enfront dels sistemes asíncrons, els quals no disposen d’un senyal de rellotge, i ho fan mitjançant l’intercanvi de senyals de control.

Figura 18. Comunicació Mestre-Esclau.

A la Figura 18 es defineix el connexionat de les línies necessàries per establir la comunicació entre un mestre i un esclau:

- MOSI (Màster-out, slave-in) per a la comunicació del mestre a l'esclau.
- MISO (Master-in, slave-out) per comunicació de l'esclau al mestre.
- SCK (Clock) senyal de rellotge enviada pel mestre.
- SS (Slave Select) aquesta línia és necessària per poder seleccionar cada dispositiu esclau connectat, per seleccionar el dispositiu amb el qual es va a realitzar la comunicació.

Per defecte el mestre manté en estat HIGH totes les línies SS. Quan el mestre vol establir comunicació amb esclau posa a LOW la línia SS corresponent, la qual cosa indica l'esclau que ha d'iniciar la comunicació.

En cada pols del senyal de rellotge, normalment en el flanc de pujada, el dispositiu mestre envia un bit a l'esclau i alhora rep un bit de l'esclau seleccionat.

La trama (les dades enviades) no segueix cap regla, és a dir, podem enviar qualsevol seqüència arbitrària de bits. Això fa que els dispositius connectats necessitin tenir pre-acordat la longitud i significat dels quals van a enviar i rebre.

Quan les dades s'envien des del mestre a un esclau, s'envien a una línia de dades anomenada MOSI, per "Màster Out / Slave In". Si l'esclau necessita enviar una resposta al mestre, el mestre continuarà generant un nombre de cicles de rellotge predisposats, i l'esclau introduirà les dades en una tercera línia de dades anomenada MISO, per "Màster In / Slave Out", com es pot veure representat a la Figura 19.

Figura 19. Esquema funcionament bus SPI.

L'electrònica requerida per implementar el bus SPI és senzilla i barata, fins i tot un únic registre de desplaçament pot ser suficient. A més, com el senyal de rellotge és proporcionada pel mestre, els esclaus ni tan sols necessiten disposar d'un rellotge propi.

3.4.3.1. Avantatges i desavantatges del bus SPI

El bus de comunicació SPI presenta un seguit d'avantatges i desavantatges.

Pel que fan els avantatges;

- Alta velocitat de transmissió (fins a 8 MHz en Arduino) i Full Duplex
- Els dispositius necessaris són senzills i barats, el que fa que estigui integrat en molts dispositius.
- Pot enviar seqüències de bit de qualsevol mida, sense dividir i sense interrupcions.

Mentre que els desavantatges que trobem, poden ser;

- Es requereix a 3 cables (SCK, MOSI i MISO) + 1 cable addicional (SS) per cada dispositiu esclau.
- Només és adequat a curta distàncies (uns 30cm).
- No es disposa de cap mecanisme de control, és a dir, no podem saber si el missatge ha estat rebut i menys si ha estat rebut correctament.
- La longitud dels missatges enviats i rebuts ha de ser coneguda per tots dos dispositius.

3.5. Interacció bus SPI amb Braccio Shield V.4

Com bé s'ha mencionat anteriorment , un dels desavantatges de fer servir el bus SPI, és la necessitat obligada de disposar de 4 pins digitals de l'Arduino (13, 12, 11 i 10) exclusivament per aquest tipus de comunicació. Arduino també disposa d'un port dedicat per la comunicació SPI anomenat ICSP, marcat en vermell a la Figura 20.

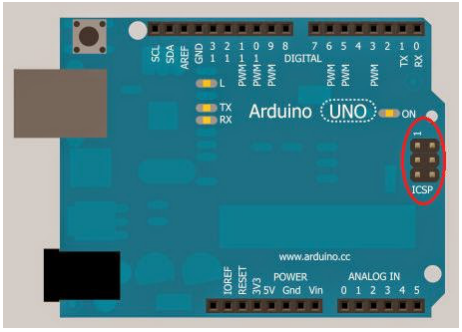


Figura 20. Imatge posició port ICSP.

El ICSP esta constituït per 6 pins assignats de la manera mostrada a la Figura 21.

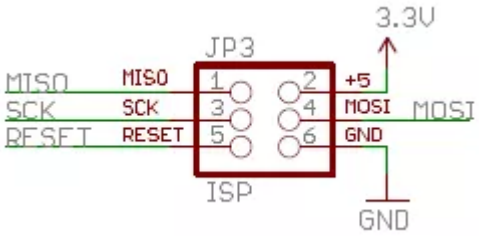


Figura 21. Esquema ICSP.

Tot i ésser un port situat fora del pinout, connecta amb el microcontrolador Atmega 328 mitjançant les potes PB5, PB4, PB3 , PB2, VCC i GND com es mostra a la Figura 22.

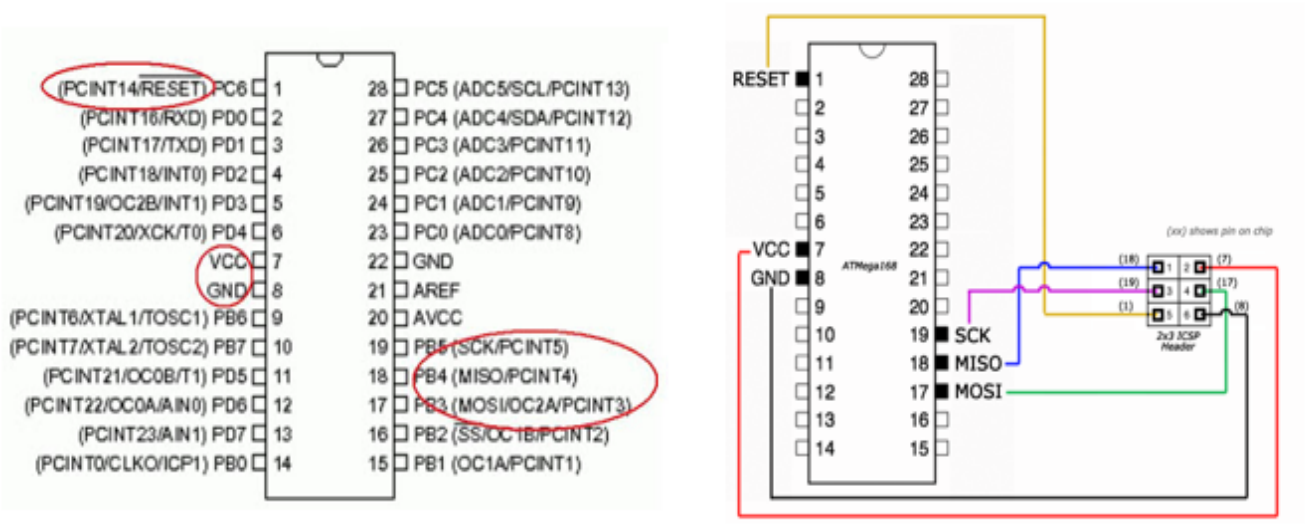


Figura 22. Assignació potes Atmega 328 amb ICSP.

Aquestes potes del microcontrolador, es comuniquen amb els pins 13, 12, 11 i 10 com ja hem mencionat anteriorment i tal i com es mostra al pinout de l'Arduino (Figura 23).

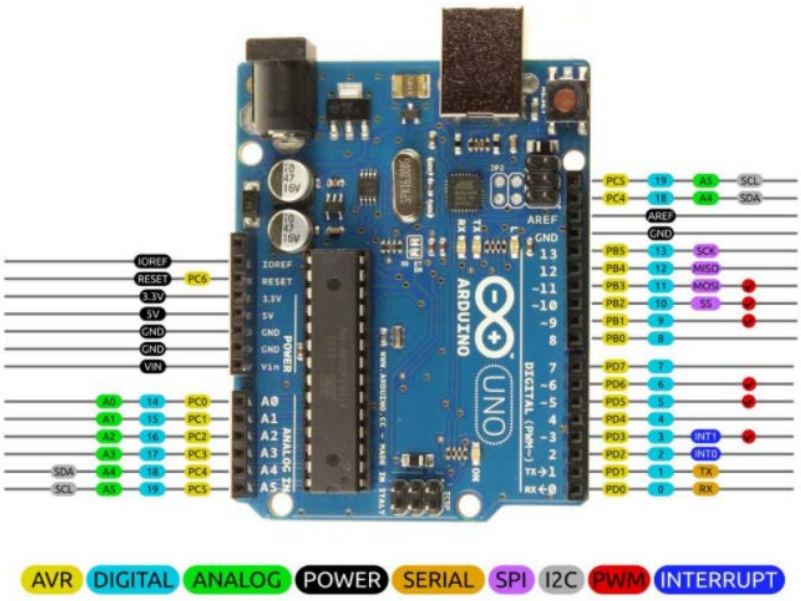


Figura 23. Pinout Arduino.

3.6. Proves, problemes i solucions

Per poder comunicar la Pixy CMUcam5 amb el Bracció Shield V.4 i fer servir el bus SPI , hem hagut de modificar l'assignació predeterminada de pins dels motors 1 i 2 tan del Braccio Shield V.4 com de les llibreries Braccio, situant aquests dos als pins lliures 4 i 2 respectivament. Aquesta modificació era necessària per poder deixar lliure els pins 11 i 10 que comandaven aquests dos motors(base.attach , shoulder.attach) i que ens impedien el correcte funcionament del bus SPI. La Figura 24 mostra el codi que permet assignar correctament els pins de comunicació amb els servo motors.

```
// initialization pin Servo motors
base.attach(4);
shoulder.attach(2);
elbow.attach(9);
wrist_rot.attach(6);
wrist_ver.attach(5);
gripper.attach(3);
```

Figura 24. Assignació pins llibreries Braccio.

La Figura 25, mostra la connexió física entre l'Arduino Uno i la Shield amb els canvis dels pins mencionats anteriorment per disposar del bus SPI.

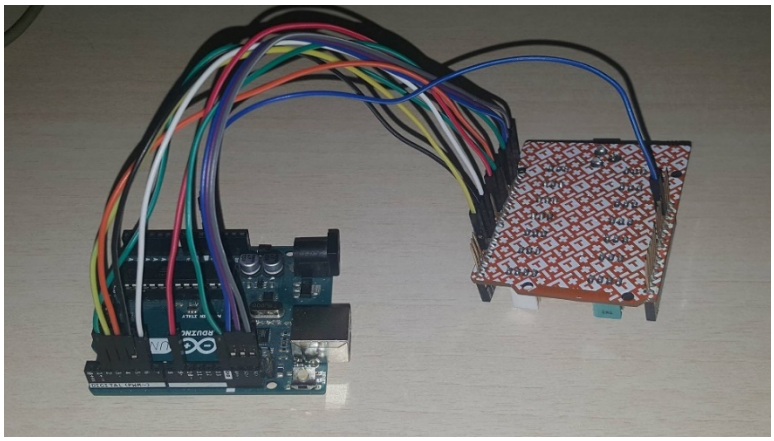


Figura 25. Nova assignació pins en la connexió Braccio Shield amb Arduino.

4. ENTORN DE PROGRAMACIÓ

4.1. Entorn Arduino

Com en qualsevol desenvolupament de software, una de les primeres eleccions és el llenguatge de programació. En el cas d'Arduino, ens centrarem en el llenguatge c / c ++, aplicat al IDE (integrated development environment) que és l'entorn de treball d'Arduino, que és el que s'ha fet servir per aquest projecte.

No obstant, existeixen altres entorns de programació, alguns més senzills i altres bastant robustos, com poden ser: Eclipse (fundation, 2018), Scracth for Arduino (S4A, 2018), Visual Studio (Microsoft, 2018) o Codebender (codeanywhere, 2018).

L'entorn de treball d'Arduino, es considera lliià ja que ens proporciona les eines bàsiques que necessitarem per carregar, depurar i comunicar-nos amb la nostra placa.

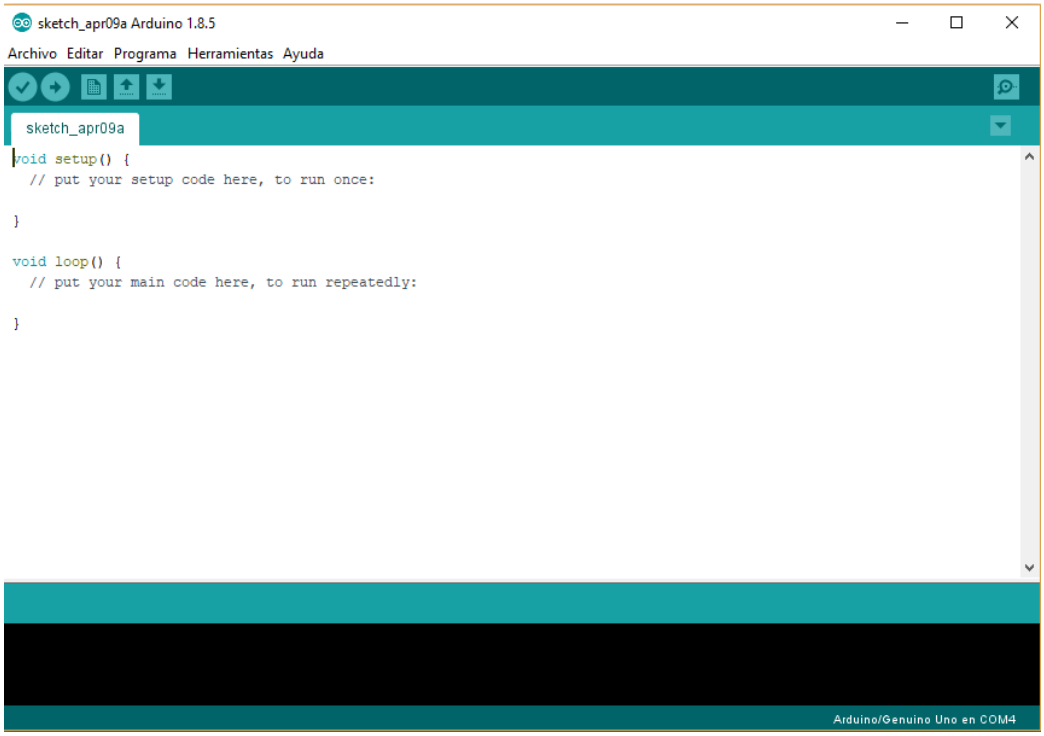


Figura 26. Entorn de programació Arduino.

De l’entorn de programació, mostrat a la (Figura 26), descriurem els menús més importants. En el menú del fitxer, trobem les següents opcions:

- **Nou Fitxer i Obrir**, respectivament ens permet crear i obrir un sketch (esbós).
- **Projecte**, ens permet obrir un conjunt de sketch que conformen un projecte determinat.
- **Exemples**, són sketch que proporcionen exemples sobre l'ús de la placa d'Arduino. Hi ha exemples bàsics que van des de fer pampallugar un led fins a fer un servidor web bàsic amb Arduino i el shield ethernet.

Del menú editar, ens dóna les opcions bàsiques d'edició (copiar, retallar i enganxar), a més de copiar el text com a html (necessari per publicar el nostre codi).

El menú programa, ens dóna les opcions de verificar i compilar el programa, incloure llibreries i mostrar la carpeta del programa, on aquesta opció és útil per poder buscar els esbossos sense tenir que navegar fins a trobar la carpeta dels fitxers localitzats a la carpeta de l'usuari, dins d'una carpeta anomenada ARDUINO. En aquesta carpeta copiarem totes les llibreries, donat que és la ruta per defecte pel que l'entorn de desenvolupament, busca els fitxers.

Les eines que ens proporciona l'IDE i que són d'interès, són les següents:

- **Auto format**, formatege el nostre codi per tal que sigui llegible. Per a projectes petits, és senzill que el programador, doni format al codi de forma senzilla, però quan té moltes línies de codi, aquesta opció és molt útil.
- **Arxiu de programa**, comprimeix tota la carpeta del projecte per guardar-la en un fitxer ZIP.
- **Reparar codificació i recarregar**, aquesta opció és útil per a reparar fitxers amb diferents codificacions. Això ens permet reparar els fitxers.
- **Monitor sèrie**, és una simple finestra, que ens permet comunicar-nos amb la placa. Ens permet tant enviar com rebre dades de l'Arduino, enviats a través de l'objecte Serial.
- **Placa i Port**, ens permet seleccionar amb quina placa estem treballant (en el nostre cas Arduino Genuino/Uno) i en quin port està connectat.
- **Programador**, en cas de disposar d'un programador de PIC's, podem seleccionar un determinat per programar el nostre xip. En la majoria dels casos amb Arduino, no ho farem servir.
- **Cremar bootloader**, ens permet tornar a carregar el bootloader (programa encarregat d'iniciar el xip i perifèrics, així com iniciar el nostre programa).

Per últim, es descriuen els botons que ens apareixen, d'esquerra a dreta sota del menú (Figura 27). Aquests botons permeten verificar, carregar programa, nou sketch, carregar sketch i guardar sketch, com s'observa a la següent figura;

Figura 27. Botonera superior entorn Arduino.

4.1.1. Elements d'un programa

Quan s'obra un programa nou se'ns mostra l'estructura següent:

```
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

Figura 28. Estructura inicial Sketch.

Tenim dues funcions que es descriuen a continuació:

- **Setup o configuració**, és una funció que només s'executa una vegada a l'inici del programa, i és aquí on hem de fer les tasques d'inicialització de variables, obrir comunicacions de canals (port sèrie, ethernet, etc ...). És a dir, és el lloc on hauríem de carregar tota la configuració i estat inicial del nostre projecte.
- **Loop,** aquesta funció en Arduino és la que s'executa un nombre infinit de vegades. En encendre l'Arduino s'executa el codi del setup i després s'entra al loop, el qual es repeteix de forma indefinida fins que s'apagui o es reinicií el microcontroladors.

Doncs bé, l'estructura bàsica d'un programa a Arduino, ha de contenir aquestes dues funcions, ja que en cas contrari, no serà possible executar un algoritme (codi). D'altre banda, l'estructura del programa també estarà formada per les funcions, llibreries, etc ... que nosaltres anirem afegint al llarg del desenvolupament del codi del nostre projecte.

4.1.2. Proves, problemes i solucions

Pel que fan les proves que s'han pogut portar a terme amb el braç robot, s'ha observat que els servo motors no venien calibrats correctament, ja que l'angle de posicionament de les articulacions del braç robot, no eres les correctes després d'haver carregat el codi de calibratge (Annex 1 - TestBraccio90º) a l'Arduino. Un cop es va identificar aquest defecte, es va procedir a realitzar un calibratge dels servo motors afectats de forma manual, descargolant els 4 cargols situats a la part posterior del servo motor per tal d'obrir-lo.

Figura 29. Part interna servo motor SR 431.

A continuació i de forma molt acurada, es va intentar corregir els graus de descalibrat mitjançant el potenciòmetre que s'observa a la Figura 29. Després de uns quants intents, es va aconseguir reduir l'error de calibratge que afectava als servomotors 2, 3 i 4.

4.2. Entorn PixyMon

PixyMon és una aplicació descarregable que es coordina amb la Pixy CMUcam5 per tal de configurar i executar allò que vulguem veure a través d’ella. Es compatible amb diverses plataformes diferents, incloses Windows, MacOs y Linus, així com altres sistemes integrats més petits com Raspberry Pi y BeagleBone Black.

Aquest entorn de treball, ens permetrà executar mètodes d’identificació que Pixy fa servir mitjançant un algoritme de filtrat basat en color per detectar objectes. Els mètodes de filtrat basats en colors són populars perquè són ràpids, eficients i relativament sòlids. La majoria de nosaltres estem familiaritzats amb RGB (vermell, verd i blau) per representar els colors. Pixy calcula el color (matís) i la saturació de cada píxel RGB del sensor d'imatge i els utilitza com a paràmetres de filtrat primaris. El to d'un objecte roman sense canvis amb els canvis d'il·luminació i exposició. Els canvis en la il·luminació i l'exposició poden tenir un efecte frustrant en els algoritmes de filtrat de color, causant que es trenquin.

L'algoritme de filtrat de Pixy és robust quan es tracta de canvis d'il·luminació i exposició.

Mitjançant PixyMon, també podrem fer que Pixy sigui capaç de recordar fins a 7 firmes de colors diferents, el que vol dir que si tens 7 objectes diferents amb colors únics, l'algoritme de filtrat de color de Pixy no tindrà problemes per identificar-los. Si es necessiten més de set colors, es poden utilitzar codis de color.

Paral·lelament al reconeixement de colors, també existeix la possibilitat que Pixy pugui trobar literalment centenars d'objectes alhora. Utilitza un algoritme de components connectats per determinar on comença un objecte i acaba un altre. És llavors que després compila les mides i les ubicacions de cada objecte i els informa a través d'una de les seves interfícies (per exemple, SCI).

Un cop finalitzada la instal·lació del PixyMon, executem el programa i podrem observar la pantalla mostrada en la Figura 30:

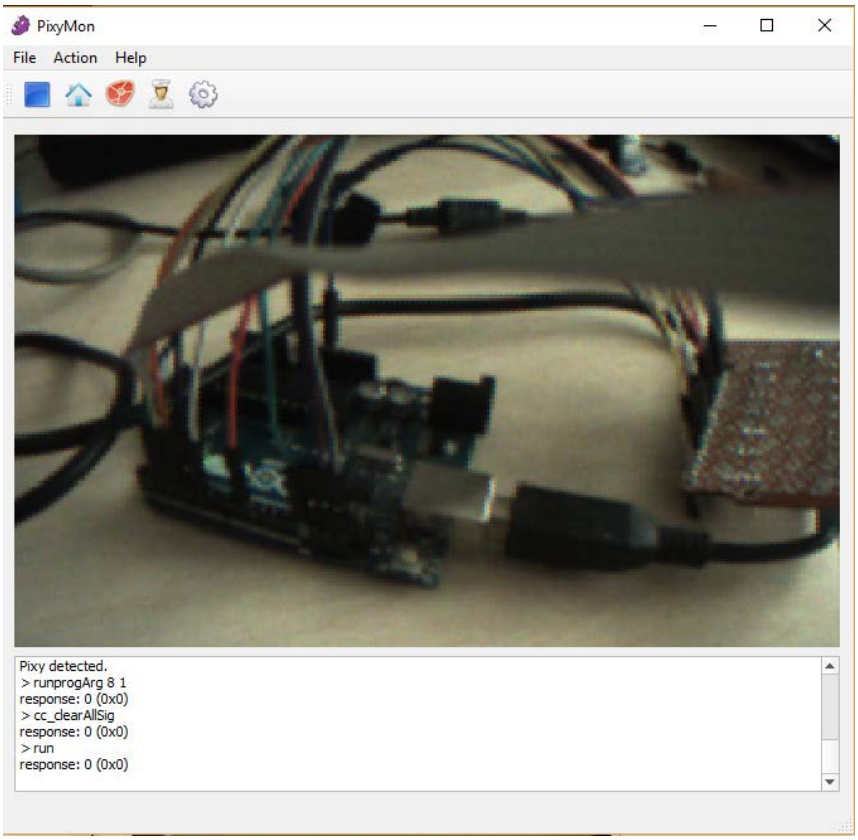


Figura 30. Entorn PixyMon.

Tal i com s'observa a la imatge anterior, l'entorn s'estructura de la següent manera:

- Botons: Estan situats a la part superior esquerra de l'entorn principal de PixyMon. Aquestes són les accions més comuns.
- Finestra de imatge: Aquí és on PixyMon renderitza diferents vídeos i ens els representa.
- Comanda/estat: A la part inferior de la finestra principal, trobem els missatges d'estat i on es poden ingressar les comandes de Pixy.

A la Figura 31 es mostren els botons disponibles de forma detallada:

Figura 31. Botonera superior entorn PixyMon.

- Parar/Iniciar: Pressionar el boto per detenir o iniciar el vídeo que s'està processant a la finestra de vídeo.
- Programa predeterminat: Executa el programa predeterminat, que és el programa que executa Pixy quan s'inicia. Normalment és el programa el que fa tot el processament en Pixy y emet els resultats (per exemple, objectes detectats) a través de uns dels ports sèrie de Pixy.
- Vídeo sense processar: Ens permet mostrar el vídeo sense processar. Això és útil per ajustar l'enfoc, la brillantor de la càmera, etc.
- Vídeo cuinat: Ens mostra el vídeo processat "cuinat". Les dades no s'envien a través dels ports sèrie de Pixy en aquest mode.
- Configurar: Pressionant aquest botó, ens mostra un altre finestra amb diferents paràmetres configurables per Pixy y PixyMon.

4.2.1. Proves, problemes i solucions

Com bé s'ha detallat a l'apartat anterior, pixy fa servir un algoritme de filtrat basat en color per detectar objectes. Degut això, s'han trobat certs problemes a l'hora de portar-ho a terme, ja que en moltes de les proves fetes s'observa que en funció dels canvis d'il·luminació que es produeix a la zona de treball de pixy mentre aquesta és executada, la identificació que fa dels objectes es veu força alterada, produint-se la pixel·lació del objecte detectat.

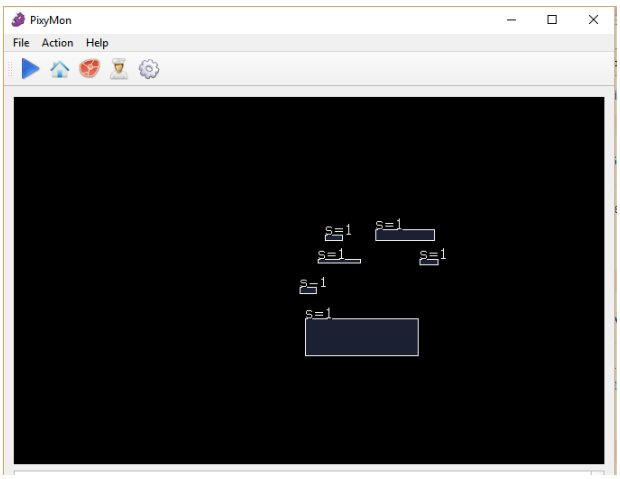


Figura 32. Pixel·lació quadrat.

És per això que per treure un rendiment més alt de la càmera, és imprescindible mantindre les mateixes condicions de lluminositat tan en el moment de guardar les firmes de color, com en el d'executar el programa predeterminat.

D'altre banda, les dades que Pixy compila dels diferents objectes com l'alçada i amplada, manquen de precisió, on les mides d'aquests, poden variar en cada identificació. Per exemple, en el cas d'un quadrat, on tots els seus costats són iguals (alçada(height)=amplada(width)), pixy pot donar petits valors d'alçada més grans que d'amplada i a l'inrevés.

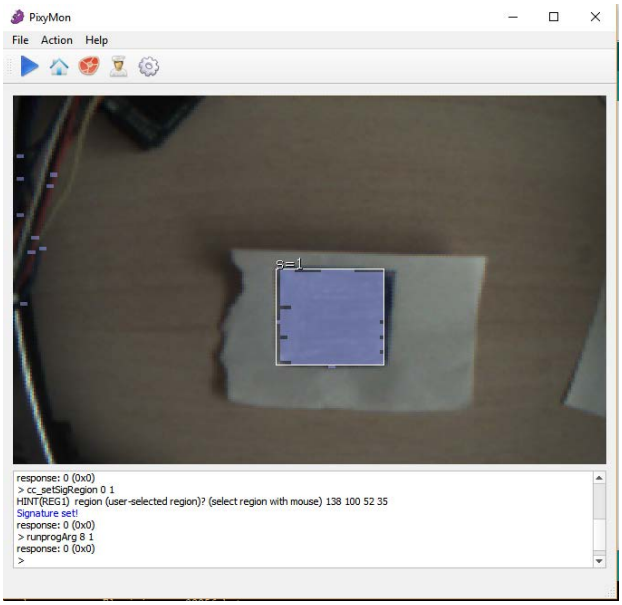


Figura 33. Identificació quadrat blau.

El que s'observa a la Figura 33, és la selecció de l'objecte en funció del seu color, on la variable S=1 (signature==1) voldrà dir que en aquest cas, el color blau s'emmagatzema com a valor 1 dins de S. En el moment de registrar altres colors, se'ls assignarà un altre número a la variable S, S=2,S=3..etc

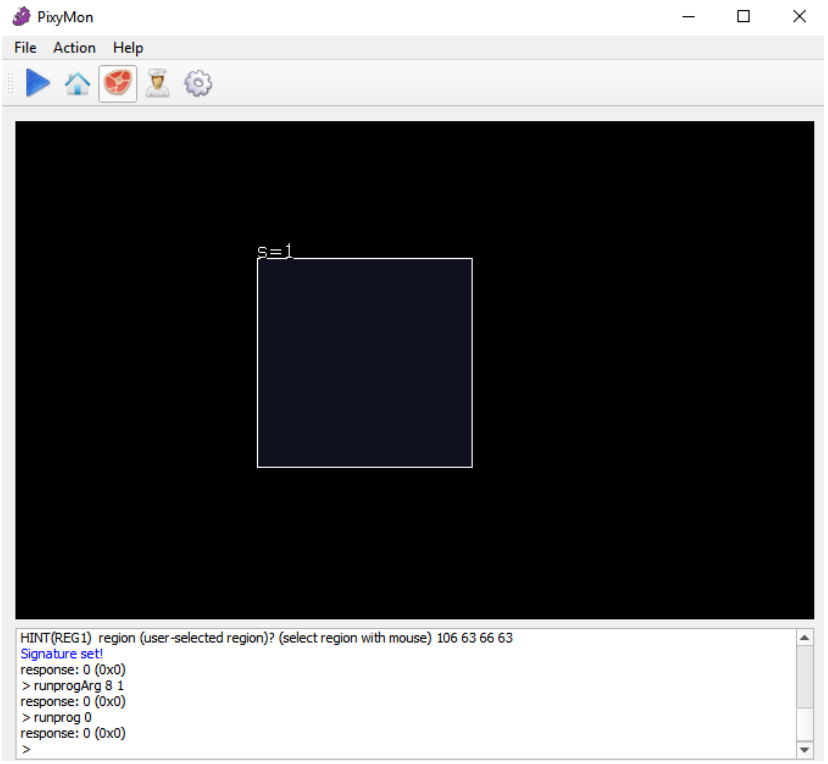


Figura 34. Execució quadrat blau.

Una vegada s'ha identificat el quadrat blau, s'executarà el programa per que Pixy pugui obtindre dades de l'objecte que està detectant (veure Figura 34).

Els paràmetres que s'obtenen del quadrat es poden veure a la Figura 35, i d'on s'han realitzat 9 identificacions per a les mateixes condicions de lluminositat, obtenint paràmetres diferents en cadascuna d'elles.

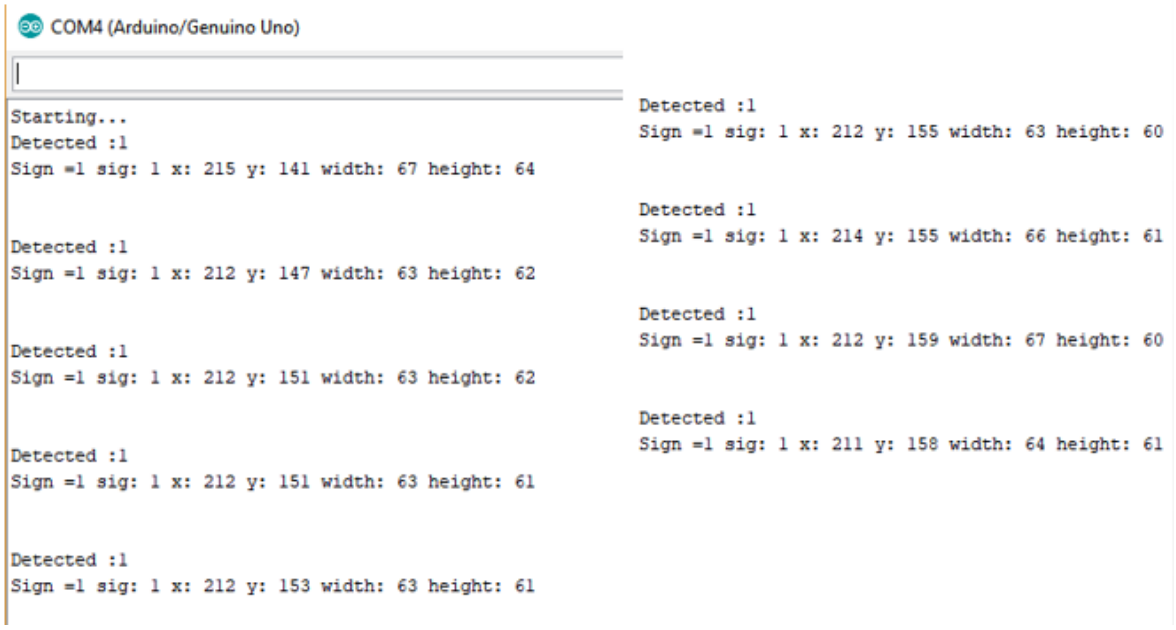


Figura 35. Paràmetres quadrat blau.

Les unitats dels valors donats, són píxels, i es veu com en aquest cas, pot arribar a representar valors de fins 7 píxels d'amplada més grans que d'alçada. Aquest marge d'error serà utilitzat en el programa desenvolupat per identificar el tipus de peça. Aquest codi es pot trobar a l'Annex 1 (ProgramaFinalArduino).

La càmera s'utilitzarà per poder reconèixer peces de característiques geomètriques diferenciades, amb la restricció de que les peces a classificar tinguin un width and height diferenciat. Aquesta condició no es dona entre un objecte quadrat, un objecte circular i un triangle rectangle de la mateixa mida.

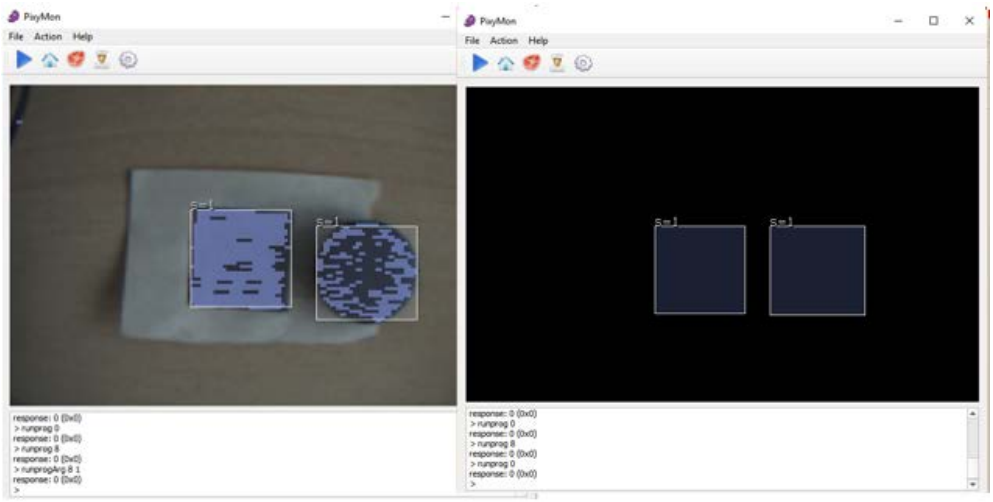


Figura 36.Comparativa quadrat-cercle.

Com s’observa a la Figura 36, quan s’executa el programa, tan el quadrat com el cercle els enquadra exteriorment obtenint paràmetres no reals en el cas del cercle, ja que el contorn d’aquest no es correspon amb l’encasellament que el programa fa de manera automàtica. El mateix succeeix amb el triangle rectangle.

És per això, que un cop s’han conegut les capacitats i limitacions que la càmera ens proporciona, s’han establert els objectes a detectar, fent servir un rectangle on l’amplada sempre serà més gran que l’alçada, un quadrat on tots els seus costats seran iguals i un triangle isòsceles, on la seva alçada serà més gran que la seva amplada (veure Figura 37).

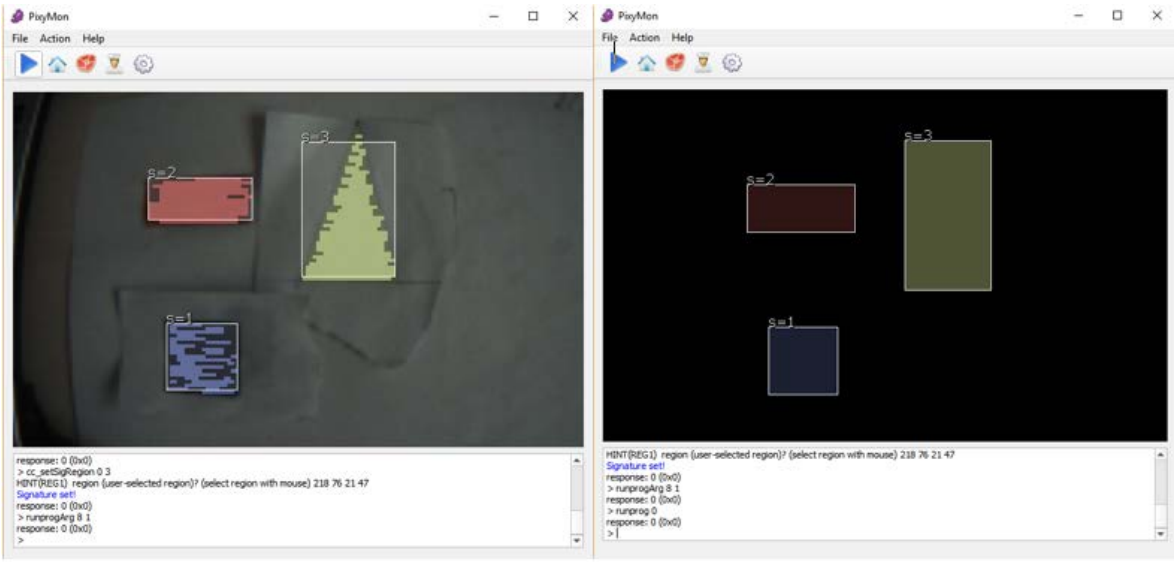


Figura 37. Identificació quadrat, rectangle i triangle.

5. UNITY ENTORN DE SIMULACIÓ 3D

5.1. Definició

Unity és un motor de videojoc multi plataforma creat per Unity Technologies (cursopedia, 2018).

És una de les plataformes per desenvolupar videojocs més complets que existeixen. Permet la creació de jocs, aplicacions interactives, visualitzacions i animacions en 3D en temps real per a múltiples plataformes a partir d'un únic desenvolupament.

L’entorn Unity 3D, ha sigut d’ajuda per tal de simular els diferents moviments que el braç robot pot realitzar, coneixent en tot moment els angles que adopten les diferents articulacions i que posteriorment s’implementaran al codi del programa per portar-ho a terme.

5.2. Entorn de simulació

L'editor d'Unity 3D és un dels més simples i potents del mercat. Es divideix en 5 vistes principals:

- **Jerarquia:** Llista jeràrquica dels elements de la escena.
- **Inspector:** Mostra i defineix les propietats dels elements del projecte. Modifica valors de forma ràpida, canvia textures arrossegant fitxers des de l'explorador, afegeix guions, guarda prefabricats, tal i com s’observa a la Figura 38 .
- **Escena:** Disseny i maqueta del joc complet o una pantalla o la secció d'aquest. Cada escena representa un nivell o secció diferent del joc (portada, nivell 1, nivell 2, inici de sessió, ...). Simplement s’arrossega els actius des de l'explorador i edita les seves variables des de l'inspector.
- **Joc:** Visualitza el teu joc a diferents resolucions. Vinculació Braç Robot amb Unity 3D.
- **Objecte de Joc:** Quan un actiu és usat en una escena de joc, es converteix en un "**Game Object**". Tot “**Game Object**” conté almenys un component amb el qual començar, és a dir, el component Transform, el qual li diu al motor d'Unity la posició, rotació, i l'escala d'un objecte.
- **Projecte:** Llista tots els elements (o actius) dels teus projectes. Permet ordenar de forma senzilla la teva aplicació. En aquesta vista hi trobem: les imatges, escenes, guions, àudios, prefabricats, textures, atles i tots els elements que es poden utilitzar.

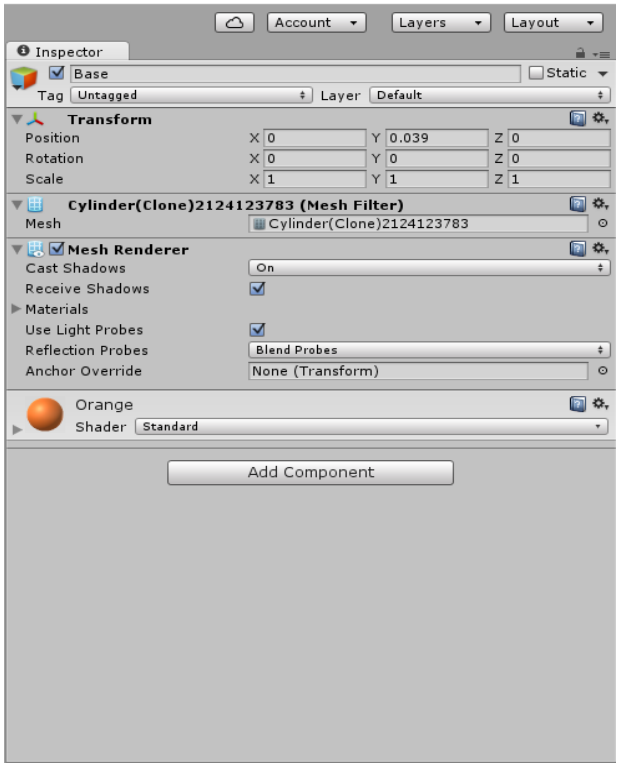


Figura 38. Inspector Unity 3D.

5.2.1. Elements d'un projecte

En el projecte cal destacar la pantalla de favorits i els assets:

- **Favorits:** Es tracta d'una pestanya on es pot guardar els arxius del treball o projecte d'ús freqüent per tal de facilitar l'accés. Es poden arrossegar els elements de la llista de jerarquía a favorits.
- **Assets:** Són blocs constructius de tot el que l'Unity posseeix en els seus treballs o projectes. Es guarden en forma d'arxius d'imatge, models del 3D, arxius de d'àudio.

Dins dels assets també hi trobem una de les funcions més interessants de l'Unity, **els Scripts**, els quals s'encarreguen de definir el comportament del joc o animació 3D.

Un Script permet modificar components d'un joc, propietats amb els temps i respondre al input de l'usuari de la forma que es desitgi mitjançant connexions amb el funcionament intern del Unity.

A diferencia d'altres assets, els Scripts es poden crear directament dins del Unity des del menú **Create.**

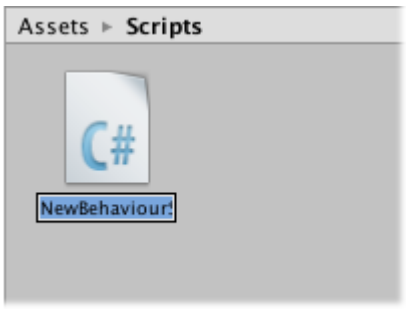


Figura 39. Creació Script.

Un cop creat el Script, a l'obrir-ho, aquest ho farà amb l'editor de text, que per defecte, Unity fa servir el MonoDevelop(entorn de desenvolupament integrat lliure i gratuït, dissenyat primordialment per a C #).

A la Figura 40 es mostra l'estructura inicials d'un arxiu Script.

```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Figura 40. Nou Script.

La funció void Start ( ) serà cridada per Unity abans que s'inicialitzi el joc o animació, mentre que la funció Update ( ), serà l'encarregada de disposar del codi que actualitzarà per frame el **GameObject**.

Un script fa les seves connexions amb el funcionament intern de Unity a l'implementar una classe que deriva des de la classe integrada anomenada MonoBehaviour. Cada vegada que s'adjunta un component script a un GameObject, aquest crea una nova instància de l'objecte definit pel pla.

Un cop s'adjunta, el Script començarà a funcionar quan es premi el botó de play, llavors el joc o animació s'engegarà.

A la Figura 41 es mostra l'entorn Unity 3D amb la simulació del braç robot Tinkerkit.

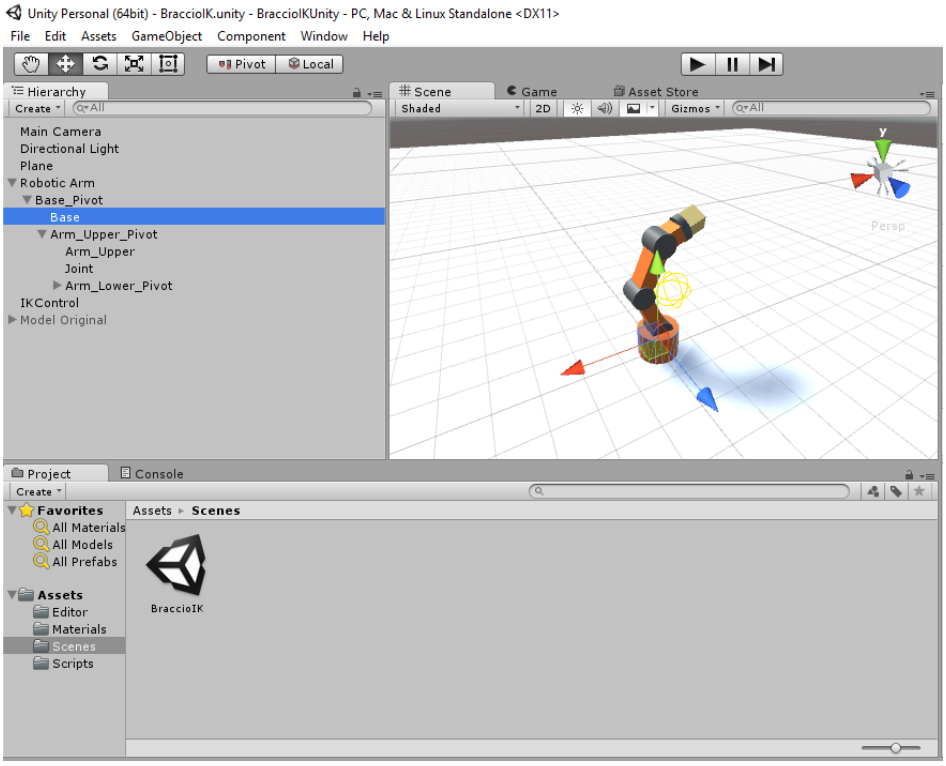


Figura 41. Entorn Unity 3D.

5.2.2. Comunicació Arduino amb Unity 3D

La comunicació entre l'Arduino Uno i l'Unity 3D, és fonamental per poder simular el braç robot i poder trobar i configurar les trajectòries que necessitarem establir més endavant al codi d'Arduino.

Primerament, l'Unity disposarà de dos Scripts ja integrats on cadascun realitzarà una funció diferent. El programa SolveIK, s'encarregarà de definir la mobilitat i els angles de cada articulació que adoptarà el braç robot, segons es representi a l'Unity 3D. D'altre banda, el segon Script d'Unity, ArduinoSerial.cs, defineix les posicions per a les diferents articulacions i al mateix temps, intentarà establir comunicació amb el propi Arduino Uno, ja connectat al PC via USB, mitjançant comunicació sèrie amb el port "COM4".

En el cas que s'aconsegueixi establir comunicació amb l'Arduino, ArduinoSerial.cs, cridarà SolveIK.cs, i s'obtindrà l'angle de posicionament de cada articulació i el qual serà enviat com bé s'ha mencionat anteriorment, cap a l'Arduino Uno.

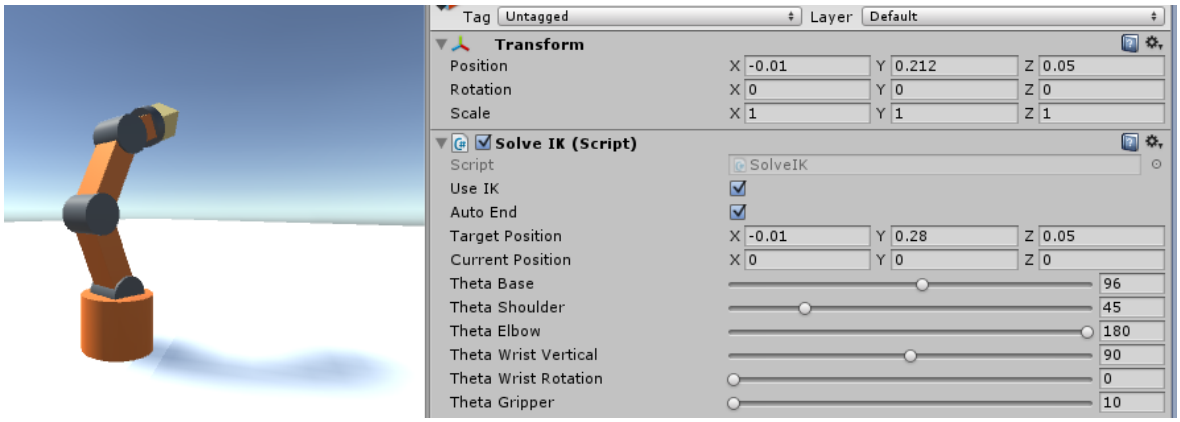


Figura 42. Posicions Angles SolveIK.

Finalment, l'Arduno Uno, tindrà carregat un Sketch amb el nom de BraccioSerialArduino, que permetrà rebre les dades enviades per l'Unity i representar-les al braç robot, descrivint els angles de cada articulació tal com s'ha establert al Unity 3D.

És per això, que la comunicació Arduino-Unity 3D és de gran ajuda, ja que et permet representar i simular el braç robot quan s'ha integrat en una estació de treball.

5.2.3. Proves, problemes i solucions

Una vegada es va instal·lar Unity3D i es van carregar els scripts i sketchs corresponents, no era possible realitzar la comunicació entre Arduino Uno i Unity 3D. El problema de comunicació era degut a que el port de comunicació no estava correctament obert.

```
void Start () {
        arduino = new SerialPort ("COM4", 9600);
        arduino.Open ();
```

Figura 43. Comunicació Serial Port.

Dins de l'arxiu ArduinoSerial.cs (Annex I), es va haver d'establir el port "COM4" del nostre ordinador, al Serial port dins del void Start tal i com s'observa a la Figura 43. Paral·lelament aquesta modificació, a l'inici del Script mencionat anteriorment, es va introduir la classe (System.IO.Ports) que ens permet iniciar les comunicacions dels ports d'entrades i sortides.

```
using System.IO.Ports;
```

Figura 44. Classe que permet iniciar comunicació dels ports.

Després d'establir la comunicació, tot ha funcionat correctament un cop executat Unity 3D.

6. DISENY, MONTATGE I EXECUCIÓ

6.1. Integració del robot amb la càmera

La integració del conjunt del hardware ho componen, la Pixy CMUcam5, l'Arduino Uno, la Braccio Shield i el braç robot (Braccio Tinkerkit). Cadascun d'ells, depèn d'un component que l'hi permet intercanviar informació amb l'ordinador.

Inicialment, disposarem de l'Arduino Uno, la qual anirà connectada al ordinador mitjançant l'extrem USB del cable d'impressora, i l'extrem del cable de connexionat d'impressora, es connectarà a l'Arduino Uno, per tal de carregar Sketchs, com es mostra a la Figura 45.

Figura 45.Cablejat Arduino Uno.

En el cas de la Pixy CMUcam5, és imprescindible connectar un cable mini USB, de la part posterior de la Pixy, al port USB de l'ordinador, ja que ens permetrà inicialitzar la càmera per poder-la fer servir amb el programa PixyMon.

D'altre banda, la càmera es connecta al capçal ICSP de l'Arduino Uno mitjançant el cable que ve amb la càmera per defecte i que s'acobla al port que I/O per a connexions SPI, UART, I2C, que es troba a la part posterior d'aquesta. Aquesta comunicació amb Arduino Uno, s'encarregarà de rebre i processar les dades que Pixy captarà.

El cablejat de la Pixy es pot observar a la següent figura:

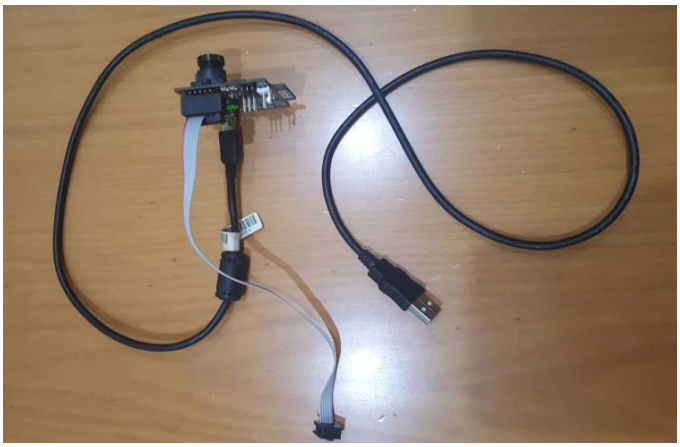


Figura 46. Cablejat PixyCMUcam5.

Finalment, pel que fa el braç robot, es connectarà amb la shield mitjançant el cablejat de cada motor ja enumerat, al seu capçal corresponent. Llavors, la Shield s'acoblarà amb l'Arduino Uno amb un connexionat que anirà des de les potes de la Shield, fins a les entrades digitals de l'Arduino com s'observa a la Figura 47, després d'haver re-assignant els pins de cada motor (vist en apartats anteriors). La Shield s'alimentarà mitjançant un transformador, que es connectarà a l'entrada de 5V.

Figura 47. Cablejat Shield-Braç Robot.

Per tant, el conjunt final on es realitzarà la integració de tots els components serà la següent:

Figura 48. Integració final del hardware.

6.2. Disseny del magatzem

El disseny del magatzem intel·ligent, es va dur a terme una vegada es va tindre clar el concepte i les tasques que realitzarien la coordinació entre el braç robot i la pixy CMUcam5. Per la realització del muntatge, es van fer servir unes làmines de fusta força robustes que permetrien establir una estructura consistent i adequada per desenvolupar aquesta activitat.

Les diferents dimensions dels materials fets servir són els següents:

- Dues làmines de fusta amb una llargada de 35,5 cm, 8 cm d'amplada i un gruix de 1,5 cm, tal i com es pot veure a la Figura 49.

Figura 49. Fusta lateral esquerra/dret magatzem intel·ligent.

- Quatre làmines de fusta amb una llargada de 23,5 cm, 7,7 cm d'amplada i un gruix de 1,5 cm, com es pot veure a la Figura 50.

Figura 50. Fusta superior, base i intermitges.

La primera part del muntatge realitzat amb el material explicat anteriorment, roman de la següent manera:

Figura 51. Primer tram del muntatge.

Per establir la configuració de la Figura 51, es van fer servir claus de 2 mm de diàmetre per tal d'acoblar les fustes laterals corresponents a la Figura 49 amb les fustes superiors, intermitges i inferiors de la Figura 50. Les distàncies que es mantenen entre làmines són de 10 cm.

Per a la realització de la segona part del muntatge, s'han fet servir:

- Dues làmines de fusta amb unes dimensions de 35,3 cm d'alçada, 26,7 cm d'amplada i un gruix de 3mm, com es veu a continuació:

Figura 52. Làmines recobriment posterior.

Aquestes dues làmines, serviran per recobrir la part posterior del primer tram del muntatge, per a cada magatzem.

Per finalitzar el conjunt, és necessari l'ús de:

- Sis làmines de fusta amb unes dimensions de 8 cm d'amplada, 10 cm d'alçada i 3 mm de gruix, com es pot veure a la Figura 53.

Figura 53. Làmines divisòries.

Aquestes làmines, complimentaran les estructures finals, fent la funció de dividir cada magatzem en 6 compartiments, tal i com es mostra a la següent imatge:

Figura 54. Muntatge final magatzems.

D'altre banda, també s'ha implementat un petit compartiment per tal de desar les peces considerades com defectuoses i que no s'adeqüen als criteris establerts.

El material fet servir per muntar-ho és el següent:

- Dues làmines de fusta amb unes dimensions de 28,5 cm d'amplada, 6,5 cm d'alçada i 1,3 cm de gruix, com es pot veure a la Figura 55.

Figura 55. Làmina d'un lateral del compartiment.

Aquestes dues làmines, aniran una a cada lateral del compartiment.

- Dues làmines de fusta amb unes dimensions de 13 cm d'amplada, 6,5 cm d'alçada i 1,3 cm de gruix, com podem veure a continuació:

Figura 56. Làmines part superior i inferior.

Les làmines de la Figura 56, conformaran el compartiment o calaix per a les peces defectuoses. L'estructura final romandrà de la següent forma (veure Figura 57):

Figura 57. Compartiment final peces defectuoses.

Cal remarcar, que l'acoblament dels materials de la Figura 52, Figura 53, Figura 55 i Figura 56, s'ha dut a terme mitjançant l'ús de cola termofusible aplicada amb una pistola. També ha sigut aplicada en espais buits entre fustes per tal de reforçar l'estructura.

Figura 58. Pistola amb cola termofusible.

6.3. Especificacions de funcionament

El magatzem intel·ligent estarà format per dos magatzem de idèntiques característiques i dimensions, els quals estaran posicionats un a cada costat del braç robot i on el compartiment de peces defectuoses, es situarà darrere d'aquest. La càmera es situarà davant del braç robot amb l'objectiu cap avall i subjectada per una petita estructura d'alumini (veure Figura 59).

Figura 59. Conjunt final.

Cada prestatge correspondrà a una peça amb un color determinat, com es mostra a la següent configuració:

Figura 60. Assignació peças magatzem 1.

Com s'observa a la imatge anterior, les peces de color blaves aniran a la part superior, posicionant-se d'esquerra a dreta, la quadrada, rectangular i triangular.

El mateix es compleix per la fila de la part inferior del magatzem 1 i mantenint el criteri establert, per al magatzem 2:

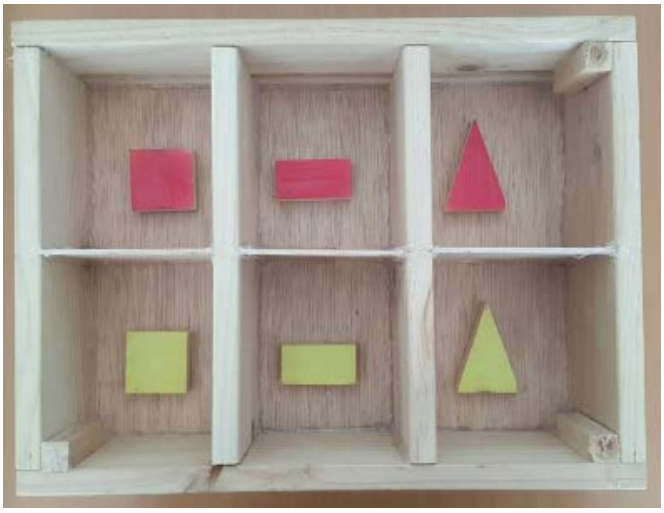


Figura 61. Assignació peces magatzem 2.

Pel que fa el compartiment de peces defectuoses, el robot hi desarà tot aquell objecte que pugui esta format per un color taronja o rosa, independentment que sigui quadrat, rectangular o triangular (veure Figura 62).

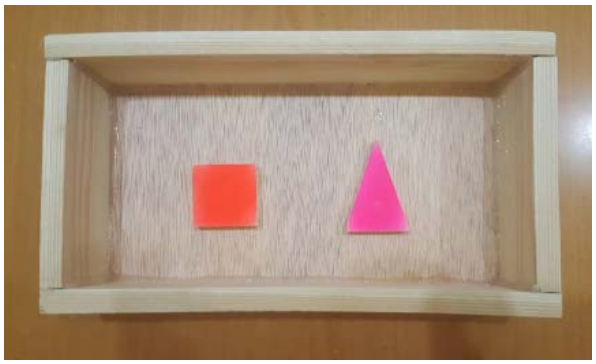


Figura 62. Compartiment peces defectuoses.

6.4. Programació

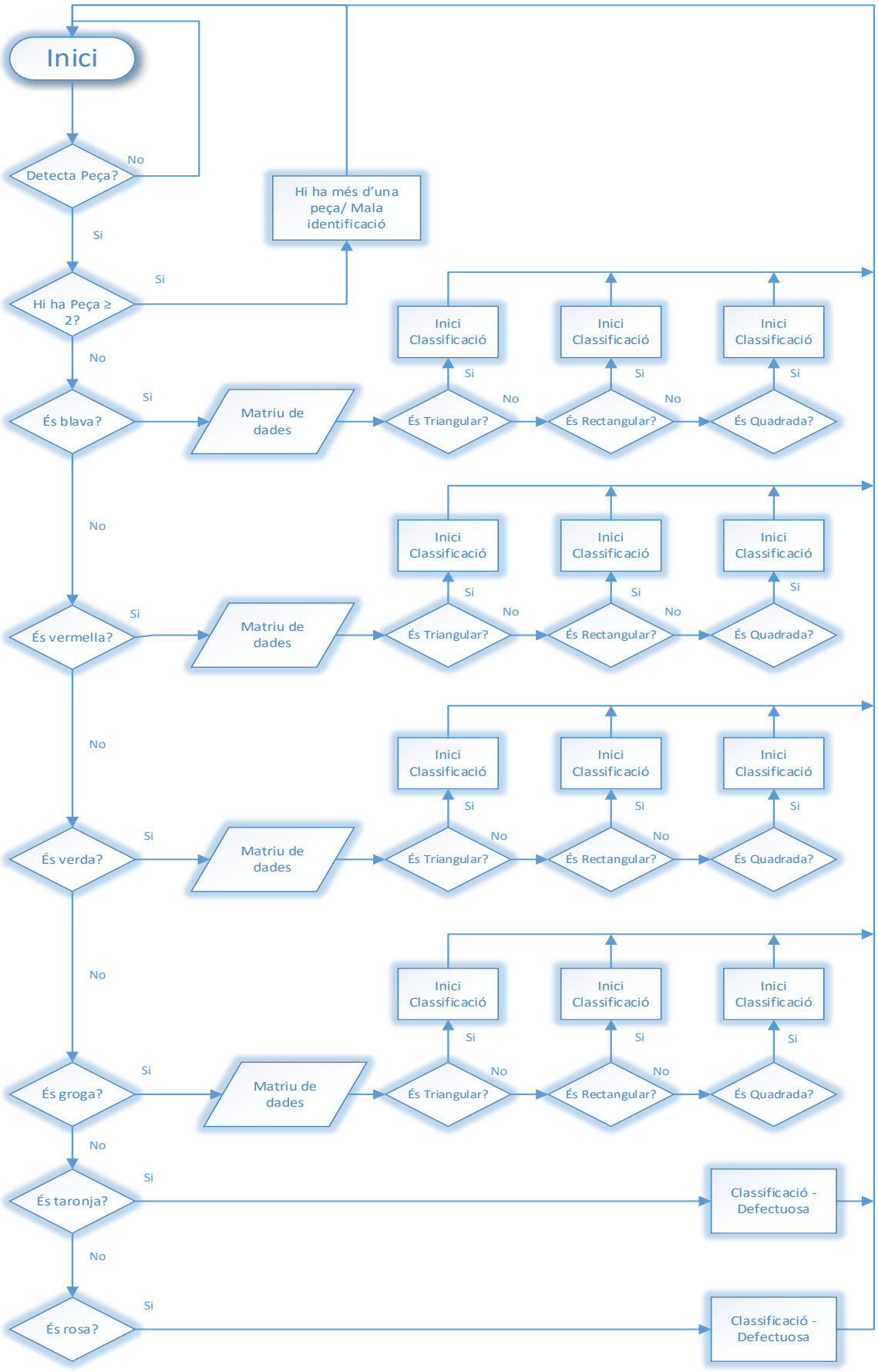
El programa s'iniciarà i la càmera comprovarà si es detecta peça. En cas que no es detecti res, aquesta continuarà comprovant fins realitzar una identificació. Si la càmera identifica un objecte, comprova si se'n ha identificat 2 o més i en cas que fos així, sortirà un missatge

per pantalla dient que s’ha detectat més d’un objecte o que s’ha produït una mala identificació, tornant a l’inici per realitzar una nova identificació.

En cas contrari, la càmera comprovarà si aquesta peça és blava i si es compleix la condició, la classificarà en funció de la seva forma en quadrada, rectangular o triangular segons la matriu de dades.

Si la identificació que fes de la peça no fos blava, continuaria comprovant per colors i posteriorment diferenciant per formes segons les matrius de dades que s’obtinguin.

En el cas de les peces taronges o roses, iniciarà la classificació sense comprovar les seves formes geomètriques.


Inici
Detecta Peça?
No
Si
Hi ha més d'una peça/ Mala identificació
Hi ha Peça ≥ 2?
Si
No
És blava?
Si
Matriu de dades
Inici Classificació
Si
És Triangular?
No
Inici Classificació
Si
És Rectangular?
No
Inici Classificació
Si
És Quadrada?
No
És vermella?
Si
Matriu de dades
Inici Classificació
Si
És Triangular?
No
Inici Classificació
Si
És Rectangular?
No
Inici Classificació
Si
És Quadrada?
No
És verda?
Si
Matriu de dades
Inici Classificació
Si
És Triangular?
No
Inici Classificació
Si
És Rectangular?
No
Inici Classificació
Si
És Quadrada?
No
És groga?
Si
Matriu de dades
Inici Classificació
Si
És Triangular?
No
Inici Classificació
Si
És Rectangular?
No
Inici Classificació
Si
És Quadrada?
No
És taronja?
Si
Classificació - Defectuosa
No
És rosa?
Si
Classificació - Defectuosa

6.4.1. Proves, problemes i solucions

Pel que fa el disseny i les especificacions de funcionament, no han sorgit problemes de caràcter greu.

Cal remarcar que per una millor visualització en el moment de la classificació, els magatzems haurien d’haver sigut d’una menor alçada o profunditat permetent també poder desar la peça amb més delicadesa. També existeix dificultat per a les pinces en agafar les peces triangulars, ja que aquestes en molts casos llisquen.

7. ESTUDI ECONÒMIC

Podem fer un estudi de costos:

Taula 3. Estudi de costos.

Component	Preu(€)
Arduino Uno	25.41 €
Braccio Tinkerkit	240 €
Pixy CMUcam5	79.50 €
Conjunt sencer fustes	10 €
Pistola encolar Salki 80W	10 €
Total	**364.91 €**

8. CONCLUSIONS

Amb la realització d'aquest projecte s'ha volgut iniciar en el món de la robòtica i automàtica. Tot i no disposar d'assignatures específiques en l'àrea de la robòtica durant el grau, s'ha pogut afrontar el projecte amb els coneixements previs d'altres assignatures amb continguts curriculars adequats, però sense dubte, també s'han hagut de solucionar certes complicacions que han anat sorgint en el decurs del treball.

Tot i existir un grau elevat d'exigència en el desenvolupament d'una tasca com aquesta, cal esmentar que hi ha prou material per intentar afrontar-ho. El gran desenvolupament de sensors, actuadors i plaques controladores permet tindre un ventall de possibilitats a l'hora de plantejar quina es la opció més òptima.

La meva proposta ha sigut intentar integrar diferent hardware controlat per un mateix software de caràcter lliure amb una funcionalitat enfocada a la classificació d'objectes. Centrant-nos en el hardware, es va escollir un braç robot estandarditzat de la marca Tinkerkit, on s'han trobat certs problemes de documentació, suport i compatibilitat. Pel que fa la càmera Pixy CMUcam5, cal esmentar que és una bona opció per iniciar-se en el món de la visió artificial ja que és present a molts projectes. També caldria remarcar, que les seves limitacions depenen del hardware que l'acompanya, ja que en aquest cas, hagués sigut adequat haver utilitzat un Arduino Mega o una Raspberry Pi B per tindre major capacitat per llegir llibreries, en comptes de l'Arduino Uno que s'ha fet servir. És per això, que per treure un major rendiment, hagués sigut adequat fer ús de les llibreries de funcions anomenades OpenCV.

El meu objectiu inicial va ésser el d'establir unes pautes de reconeixement d'objectes segons formes i colors. La problemàtica resideix en que les limitacions de la càmera mencionades anteriorment, impossibilitaven assolir-ho completament. És per això, que s'ha treballat i obtingut uns objectius, tenint en compte aquestes restriccions, podent realitzar una classificació segons colors i buscant la manera de poder diferenciar entre 3 figures geomètriques.

Per concloure, es podria dur a terme o continuar amb un projecte semblant al realitzat, però fent ús de productes de nova implantació de la casa Arduino, afegint una petita funcionalitat del PLC de l'estació de treball i millorar el nivell de programació per establir un procés més compacte.

9. BIBLIOGRAFIA

Arduino. (20 de 2 de 2018). *Arduino*. Obtenido de https://www.arduino.cc/

BricoGeek. (16 de 2 de 2018). *BricoGeek*. Obtenido de http://tienda.bricogeek.com/sensores-imagen/755-camara-pixy-cmucam5.html

codeanywhere, C. a. (10 de 4 de 2018). *Cobender*. Obtenido de https://codebender.cc/

cursopedia. (27 de 2 de 2018). *Cursopedia*. Obtenido de http://www.cursopedia.com/Ficha-Unity-3D-para-principiantes

fundation, E. (10 de 3 de 2018). *ECLIPSE fundation*. Obtenido de https://www.eclipse.org/community/eclipse_newsletter/2017/april/article4.php

Llamas, L. (28 de 2 de 2018). *Luis Llamas*. Obtenido de https://www.luisllamas.es/arduino-spi/

Microsoft. (10 de 4 de 2018). *Microsoft*. Obtenido de https://blogs.msdn.microsoft.com/iotdev/2017/07/06/visual-studio-code-extension-for-arduino-is-now-open-sourced/

Pixy, C. (20 de 2 de 2018). *Cmucam5 Pixy*. Obtenido de http://cmucam.org/projects/cmucam5/wiki/Uploading_New_Firmware

R3D*. (25 de 2 de 2018). *R3D**. Obtenido de http://r3dstar.co.uk/?p=211

Robótica, P. y. (18 de 2 de 2018). *Programación y Robótica*. Obtenido de http://www.programacionyrobotica.com/practicas-arduino/

RS-Online. (15 de 2 de 2018). *rs-online*. Obtenido de https://www.rs-online.com/designspark/building-braccio-the-tinkerkit-robot-arm

S4A. (10 de 4 de 2018). *S4A*. Obtenido de http://s4a.cat/

t21mx. (15 de 5 de 2018). *t21mx*. Obtenido de http://t21.com.mx/logistica/2012/09/20/beneficios-almacenes-inteligentes

10. ANNEXOS

Inicialment, per a la familiarització amb el programa Unity 3D, es van realitzar diverses pràctiques per compte propi abans de fer servir la simulació del braç Robot (R3D*, 2018). El mateix per iniciar-me amb l'Arduino Uno, servint d'ajuda per poder començar el projecte amb una idea prèvia del seu funcionament (Robótica, 2018).

10.1. Annex 1

En aquest apartat, es presenten els diferents programes fets servir al projecte:

Codis Arduino

- **TestBraccio90**

Aquest és un sketch de configuració i per comprovar que els servomotors estiguin completament ben alineats.

```
//S'estableixen les llibreries dels sevomotors i del braç robot.
#include <Braccio.h>
#include <Servo.h>

//Es defineixen les articulacions del braç.
Servo base;
Servo shoulder;
Servo elbow;
Servo wrist_rot;
Servo wrist_ver;
Servo gripper;


//Es defineixen les funcions d'inicialització, es configura la posició inicial del braç i tots els servomotors s'establiran en la posició de "seguretat".
void setup() {

  Braccio.begin();
 }

void loop() {

Braccio.ServoMovement(20,         90, 90, 90, 90, 90,  73);
```

```
  delay(2000);

}
```

- **ProgramaFinalArduino**

```
#include <SPI.h>
#include <Pixy.h>          // Llibreries que es fan servir
#include <Braccio.h>
#include <Servo.h>
Servo base;                // Indiquem els diferents servos que farem servir.
Servo shoulder;
Servo elbow;
Servo wrist_ver;
Servo wrist_rot;
Servo gripper;

Pixy pixy;


void setup()
{
  Serial.begin(9600);
  Braccio.begin();              // Iniciem port serie
  Serial.print("Starting...\n");
  pixy.init();
}



void loop()

{
  uint16_t type;
  static int i = 0;
  int j;                    // Es creen diferents variable dels tipus enter
  int v;
  int width;
  int height;
  int16_t signature;
  int16_t angle;
```

```
uint16_t blocks;
int a;

blocks = pixy.getBlocks();     // Dades que proporciona Pixy

if (blocks)

{

  i++;

  if (i % 200 == 0)

  {

    for (v =0 ;v==0; v++)     // Es crea una nova variable V inicialitzant-la en 0, per crear el bucle

    {
      Serial.print("Detected :");
      Serial.print(blocks);
      Serial.print("\n");
      Serial.print("Sign =");
      signature = (pixy.blocks[j].signature);   // Variable color
      width=(pixy.blocks[j].width);             // Variable amplada
      height=(pixy.blocks[j].height);           // Variable alçada
      angle =(pixy.blocks[j].angle);            // Variable angle
      a=width-height;
      Serial.print(signature,OCT);
      Serial.print("\t");
      pixy.blocks[j].print();                   // Escriu tots els paràmetres en una sola línia




    if (blocks>=2)            // Si detecta que hi han 2 o més peces alhora, mostra per pantalla un error dient que s'ha detectat 2 o més peces/ o bé detecta varies peces per una mala identificació de la càmera (mal posicionament)
    {
      delay(50);
      Serial.print("\n");
```

```
        Serial.print("ERROR, S'HAN DETECTAT 2 O MES PEÇAS/ MAL POSICIONAMENT DE LA PEÇA ");
        Serial.print("\n");
        break;
        delay(500);
      }

      if (signature==1)   // Si és una peça de color blau.

      {
        if ((a)<(-10))     // Si el valor de "a" = width-height, es menor que -2, es una peça Triangular.
        {
        Serial.print("\n");
        Serial.print("Peça blava amb forma Triangular detectada, inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,57,30);    //Posició inicial.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,57,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,57,85);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,57,85);       //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,42.4,130.3,128.3,158.5,57,85);   //Arriba al magatzem.

        Braccio.ServoMovement(20,42.4,130.3,128.3,158.5,57,30);     //Deixa anar la peça.
        delay(100);
        Braccio.ServoMovement(20,42.4,50,128.3,158.5,57,30);        // Puja braç
        delay(1000);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);        //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
        else if ((a)>13)   // Si el valor de "a" = width-height, es més gran que 13, es una peça Rectangular.
        {
        Serial.print("\n");
```

```
        Serial.print("Peça blava amb forma Rectangular detectada,inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,160,30);    //Posició inicial.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76,22.9,18.1,160,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,160,85);
        delay(1000);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,160,85);      //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,26.3,99.6,152.5,163.8,160,85);   //Arriba al magatzem.

        Braccio.ServoMovement(20,26.3,99.6,152.5,163.8,160,30);     //Deixa anar la peça.
        delay(1000);
        Braccio.ServoMovement(20,26.3,50,152.5,163.8,160,30);       // Puja braç
        delay(100);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
        else   // En cas contrari, és una peça Quadrada.
        {
        Serial.print("\n");
        Serial.print("Peça blava amb forma Quadrada detectada,inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,85);
```

```
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,85);      //Agafa la primera peça.
        delay(500);
        Braccio.ServoMovement(20,0,86.6,176.6,158.5,50,85);      //Arriba al magatzem.

        Braccio.ServoMovement(20,0,86.6,176.6,158.5,50,30);      //Deixa anar la peça.
        delay(1000);
        Braccio.ServoMovement(20,42.4,50,128.3,158.5,50,30);      // Puja braç
        delay(100);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);      //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
      }

        if (signature==2)
       {
        if ((a)<(-10))
        {
        Serial.print("\n");
        Serial.print("Peça vermella amb forma Triangular detectada,inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,57,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,57,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,57,85);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,57,85);      //Agafa la primera peça.
        delay(500);
        Braccio.ServoMovement(20,118.2,120,140.3,143.1,96,85);      //Arriba al magatzem.

        Braccio.ServoMovement(20,118.2,120,140.3,143.1,96,30);      //Deixa anar la peça.
        delay(100);
        Braccio.ServoMovement(20,28.3,50,122.8,108.2,96,30);      // Puja braç
        delay(1000);
```

```
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);      //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
        else if ((a)>13)
        {
        Serial.print("\n");
        Serial.print("Peça vermella amb forma Rectangular, inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,160,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76,22.9,18.1,160,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,160,85);
        delay(1000);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,160,85);      //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,133.6,96.8,157.8,158.5,96,85);                //Arriba al magatzem.

        Braccio.ServoMovement(20,133.6,96.8,157.8,158.5,96,30);     //Deixa anar la peça.
        delay(1000);
        Braccio.ServoMovement(20,14.8,50,123.4,146.3,94.7,30);      // Puja braç
        delay(100);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);      //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
        else
        {
        Serial.print("\n");
        Serial.print("Peça vermella amb forma Quadrada detectada,inici de la classificació :");
        Serial.print("\n");
```

```
        pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,85);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,85);      //Agafa la primera peça.
        delay(500);
        Braccio.ServoMovement(20,162.5,72.6,176,158.5,96,85);      //Arriba al magatzem.

        Braccio.ServoMovement(20,162.5,72.6,176,158.5,96,30);      //Deixa anar la peça.
        delay(1000);
        Braccio.ServoMovement(20,0,50,120.2,147.1,94.7,85);        // Puja braç
        delay(100);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
      }

      if (signature==3)
      {
        if ((a)<(-10))
        {
        Serial.print("\n");
        Serial.print("Peça verda amb forma Triangular detectada,inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,57,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,57,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,57,85);
```

```
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,57,85);       //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,28.3,147.7,122.8,108.2,96,85);                  //Arriba al
magatzem.

        Braccio.ServoMovement(20,28.3,147.7,122.8,108.2,96,30);     //Deixa anar la peça.
        delay(100);
        Braccio.ServoMovement(20,28.3,50,122.8,108.2,96,30);        // Puja braç
        delay(1000);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició
inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
        else if ((a)>13)
        {
        Serial.print("\n");
        Serial.print("Peça verda amb forma Rectangular, inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,160,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76,22.9,18.1,160,30);       //Baixa per recollir una
peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,160,85);
        delay(1000);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,160,85);       //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,14.8,142.7,123.4,146.3,94.7,85);          //Arriba al
magatzem.

        Braccio.ServoMovement(20,14.8,142.7,123.4,146.3,94.7,30);          //Deixa anar la
peça.
        delay(1000);
        Braccio.ServoMovement(20,14.8,50,123.4,146.3,94.7,30);             // Puja braç
```

```
        delay(100);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
        else
        {
        Serial.print("\n");
        Serial.print("Peça verda amb forma Quadrada detectada,inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,85);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,85);       //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,0,139.9,120.2,147.1,94.7,85);     //Arriba al magatzem.

        Braccio.ServoMovement(20,0,139.9,120.2,147.1,94.7,30);      //Deixa anar la peça.
        delay(1000);
        Braccio.ServoMovement(20,0,50,120.2,147.1,94.7,85);         // Puja braç
        delay(100);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
      }
      if (signature==4)
      {
        if ((a)<(-10))
        {
        Serial.print("\n");
        Serial.print("Peça groga amb forma Triangular detectada,inici de la classificació :");
        Serial.print("\n");
```

```
        pixy.blocks[j].print();
        Serial.print("\n");
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,57,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,57,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,57,85);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,57,85);       //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,135.8,134.6,141.1,106.1,8.8,85);          //Arriba al magatzem.

        Braccio.ServoMovement(20,135.8,134.6,141.1,106.1,8.8,30);          //Deixa anar la peça.
        delay(100);
        Braccio.ServoMovement(20,28.3,50,122.8,108.2,96,30); // Puja braç
        delay(1000);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
        else if ((a)>13)
        {
        Serial.print("\n");
        Serial.print("Peça groga amb forma Rectangular, inici de la classificació :");
        Serial.print("\n");
         pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,160,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76,22.9,18.1,160,30);       //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,160,85);
        delay(1000);
```

```
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,160,85);       //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,149.8,127.1,130.4,143.7,8.8,85);          //Arriba al
magatzem.

        Braccio.ServoMovement(20,149.8,127.1,130.4,143.7,8.8,30);          //Deixa anar la
peça.
        delay(1000);
        Braccio.ServoMovement(20,14.8,50,123.4,146.3,94.7,30); // Puja braç
        delay(100);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició
inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
        else
        {
        Serial.print("\n");
        Serial.print("Peça groga amb forma Quadrada detectada,inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        Serial.print("\n");
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,30);      //Baixa per recollir una
peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,85);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,85);       //Agafa la peça.
        delay(500);
        Braccio.ServoMovement(20,165.9,124.8,126.4,166.5,8.8,85);          //Arriba al
magatzem.

        Braccio.ServoMovement(20,165.9,124.8,126.4,166.5,8.8,85);          //Deixa anar la
peça.
        delay(1000);
        Braccio.ServoMovement(20,0,50,120.2,147.1,94.7,85); // Puja braç
        delay(100);
```

```
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
        }
    }
        if ((signature==5) or (signature==6))
        {
        Serial.print("\n");
        Serial.print("Peça defectuosa, inici de la classificació :");
        Serial.print("\n");
        pixy.blocks[j].print();
        delay(500);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,30);    //Posició inicial.
        delay(1000);

        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,30);      //Baixa per recollir una peça.
        delay(1000);
        Braccio.ServoMovement(20,76.5,76.5,22.9,18.1,50,85);

        Braccio.ServoMovement(20,76.5,124,22.9,18.1,50,85);      //Agafa la primera peça.
        delay(500);
        Braccio.ServoMovement(20,127.5,29.3,73.2,7.6,8.8,85);      //Arriba al magatzem.

        Braccio.ServoMovement(20,127.5,29.3,73.2,7.6,8.8,30);      //Deixa anar la peça.
        delay(1000);
        Braccio.ServoMovement(20,0,50,120.2,147.1,94.7,85);      // Puja braç
        delay(100);
        Braccio.ServoMovement(20,76.5,124,22.9,18.1,15.4,30);       //Torna a la posició inicial esperant unaltre peça.(La espera dependrà de la detecció que faci la càmera)
        delay(3000);
       }

     }
     Serial.print("\n");
   }

  }
}
```

- **BraccioSerialArduino**

```
// Afegeix les llibreries necessàries
#include <Braccio.h>
#include <Servo.h>

Servo base;
Servo shoulder;
Servo elbow;
Servo wrist_rot;
Servo wrist_ver;
Servo gripper;

int stepDelay = 20;
int m1 = 90;
int m2 = 45;
int m3 = 180;
int m4 = 180;
int m5 = 90;
int m6 = 10;

char inData[25];
bool isRead = false;
int index = 0;


//  Funcions d'inicialització i configura la posició inicial per a Braccio
void setup() {

  Braccio.begin();
  Serial.begin(9600);
}

// Tots els servomotors es posicionaran en la posició de "seguretat":
void loop() {


  readSerialStr();

  Braccio.ServoMovement(stepDelay, m1, m2, m3, m4, m5, m6);

  delay(100);
}

void readSerialStr() {
      Serial.println("Serial1");
  if (Serial.available() > 0) {
    Serial.println("Serial2");
```

```
    char incomingByte = Serial.read();
    while (incomingByte != '\n' && isDigit(incomingByte)) {
      Serial.println("Serial3");
      Serial.println(incomingByte);
      isRead = true;
      delay(100);
      inData[index] = incomingByte;
      index++;
      incomingByte = Serial.read();
    }
    inData[index] = '\0';
  }

// Lectura de dades a cada motor
  if (isRead) {
    char m1_char[4];
    char m2_char[4];
    char m3_char[4];
    char m4_char[4];
    char m5_char[4];
    char m6_char[4];

    m1_char[0] = inData[0];
    m1_char[1] = inData[1];
    m1_char[2] = inData[2];
    m1_char[3] = '\0';
    m2_char[0] = inData[3];
    m2_char[1] = inData[4];
    m2_char[2] = inData[5];
    m2_char[3] = '\0';
    m3_char[0] = inData[6];
    m3_char[1] = inData[7];
    m3_char[2] = inData[8];
    m3_char[3] = '\0';
    m4_char[0] = inData[9];
    m4_char[1] = inData[10];
    m4_char[2] = inData[11];
    m4_char[3] = '\0';
    m5_char[0] = inData[12];
    m5_char[1] = inData[13];
    m5_char[2] = inData[14];
    m5_char[3] = '\0';
    m6_char[0] = inData[15];
    m6_char[1] = inData[16];
    m6_char[2] = inData[17];
    m6_char[3] = '\0';

    m1 = atoi(m1_char);
    m2 = atoi(m2_char);
    m3 = atoi(m3_char);
    m4 = atoi(m4_char);
    m5 = atoi(m5_char);
```

```
    m6 = atoi(m6_char);

    isRead = false;
    index = 0;
  }

}
```

Scripts Unity3D

- **SolveIK.cs**

```
using UnityEngine;
using System.Collections;

public class SolveIK : MonoBehaviour {

    public bool useIK = true;  //Mode IK(inv. Kinematic) o ajust manual
    public bool autoEnd = true; //final horitzontal en mode IK

    public Vector3 targetPosition;
    public Vector3 currentPosition;

    [Range(0.0f, 180.0f)]
    public float thetaBase = 90f;
    [Range(15.0f, 165.0f)]
    public float thetaShoulder = 45f;
    [Range(0.0f, 180.0f)]
    public float thetaElbow = 180f;
    [Range(0.0f, 180.0f)]
    public float thetaWristVertical = 90f;
    [Range(0.0f, 180.0f)]
    public float thetaWristRotation = 0f;
    [Range(10.0f, 73.0f)]
    public float thetaGripper = 10f;

    public GameObject[] arms = new GameObject[5];

// Dimensions del braç (m) //
    float BASE_HGT = 0.078f;
    float HUMERUS = 0.124f;
    float ULNA = 0.124f;
    float GRIPPER = 0.058f;

// pre-calculs //
    float hum_sq;
    float uln_sq;

    void Start () {
```

```
        hum_sq = HUMERUS*HUMERUS;
        uln_sq = ULNA*ULNA;
    }

    void Update () {

        // Estableix quina serà la seva posició
        targetPosition = transform.position;

        if (useIK) {
            SetArm (targetPosition.x, targetPosition.y, targetPosition.z, autoEnd);
        }

        // Actualitza el model de braç robot
        arms [0].transform.localRotation = Quaternion.Euler(new Vector3 (0f, thetaBase, 0f));
        arms [1].transform.localRotation = Quaternion.Euler(new Vector3 (0f, 0f, thetaShould
er - 90f));
        arms [2].transform.localRotation = Quaternion.Euler(new Vector3 (0f, 0f, thetaElbow -
 90f));
        arms [3].transform.localRotation = Quaternion.Euler(new Vector3 (0f, 0f, thetaWristV
ertical - 90f));
        arms [4].transform.localRotation = Quaternion.Euler(new Vector3 (0f, thetaWristRotat
ion, 0f));

        // Posició actual del canell
        currentPosition = arms [3].transform.position;
    }

    void SetArm(float x, float y, float z, bool endHorizontal) {
        // Angle de la base
        float bas_angle_r = Mathf.Atan2( x, z );
        float bas_angle_d = bas_angle_r * Mathf.Rad2Deg + 90f;

        float wrt_y = y - BASE_HGT; //Alçada relativa del canell al hombro
        float s_w = x * x + z * z + wrt_y * wrt_y; // Distància al quadrat del hombro fins el
canell
        float s_w_sqrt = Mathf.Sqrt (s_w);

        //Angle del colze: 3 arestes per conèixer el triangle.
            float elb_angle_r = Mathf.Acos ((hum_sq + uln_sq - s_w) / (2f * HUMERUS * UL
NA));
        float elb_angle_d = 270f - elb_angle_r * Mathf.Rad2Deg;

        // Angle del hombro = a1 + a2
        float a1 = Mathf.Atan2 (wrt_y, Mathf.Sqrt (x * x + z * z));
        float a2 = Mathf.Acos ((hum_sq + s_w - uln_sq) / (2f * HUMERUS * s_w_sqrt));
        float shl_angle_r = a1 + a2;
        float shl_angle_d = 180f - shl_angle_r * Mathf.Rad2Deg;

        //Es manté el punt final en horitzontal
        if (endHorizontal) {
            float end_x = arms [4].transform.position.x;
```

```
            float end_y = arms [4].transform.position.y;
            float end_z = arms [4].transform.position.z;

            float end_last_angle = thetaWristVertical;

            float dx = end_x - x;
            float dz = end_z - z;

            float wrt_angle_r = Mathf.Atan2 (end_y - y, Mathf.Sqrt (dx * dx + dz * dz));
            float wrt_angle_d = end_last_angle + wrt_angle_r * Mathf.Rad2Deg;

            // S'actualitzen els angles
            if (wrt_angle_d >= 0f && wrt_angle_d <= 180f)
                thetaWristVertical = wrt_angle_d;
        }

        // S'actualitzen els angles
        if (bas_angle_d >= 0f && bas_angle_d <=180f)
            thetaBase = bas_angle_d;
        if (shl_angle_d >= 15f && shl_angle_d <=165f)
            thetaShoulder = shl_angle_d;
        if (elb_angle_d >= 0f && elb_angle_d <=180f)
            thetaElbow = elb_angle_d;
    }
}
```

- **ArduinoSerial.cs**

```
// S'estableixen les classes tan per obrir ports, crear matrius, fer servir funcions...
using UnityEngine;
using System.Collections;
using System.IO.Ports;

// Estableix connexió amb el port Serie.
public class ArduinoSerial : MonoBehaviour {
        public SolveIK solveIK;
        public int delaySeconds = 3;
        public string portName;
        SerialPort arduino;
        bool startCommands = false;

        void Start () {
                arduino = new SerialPort ("COM4", 9600);
                arduino.Open ();

        }
        // Comprova que estigui rebent ordres
        void Update () {
                if (startCommands == false)
```

```
                StartCoroutine (SendCommands ());
        }

        IEnumerator SendCommands () {
                startCommands = true;
                yield return new WaitForSeconds(delaySeconds);


                // Si es comunica amb Arduino, llavors estableix posicions (cadenes
de caràcters) cridant SolveIK.
                if (arduino.IsOpen) {
                        string str;

                        string thetaBaseStr =
(Mathf.RoundToInt(solveIK.thetaBase)).ToString("000");
                        string thetaShoulderStr =
(Mathf.RoundToInt(solveIK.thetaShoulder)).ToString("000");
                        string thetaElbowStr =
(Mathf.RoundToInt(solveIK.thetaElbow)).ToString("000");
                        string thetaWristVerticalStr =
(Mathf.RoundToInt(solveIK.thetaWristVertical)).ToString("000");
                        string thetaWristRotationStr =
(Mathf.RoundToInt(solveIK.thetaWristRotation)).ToString("000");
                        string thetaGripperStr =
(Mathf.RoundToInt(solveIK.thetaGripper)).ToString("000");

                        str = thetaBaseStr + thetaShoulderStr + thetaElbowStr +
thetaWristVerticalStr + thetaWristRotationStr + thetaGripperStr + "\n";

                        arduino.Write (str);

                        Debug.Log ("Send Serial: " + str);
                }
                startCommands = false;

        }
}
```

10.2. Annex 2

Les llibreries fetes servir al projecte, com són las del Braccio i la dels Servomotors:

- **Braccio.h**

```
#ifndef BRACCIO_H_
#define BRACCIO_H_

#include <Arduino.h>
#include <Servo.h>

// You should set begin(SOFT_START_DISABLED) if you are using the Arm Robot shield V1.6
#define SOFT_START_DISABLED          -999

//The default value for the soft start
#define SOFT_START_DEFAULT           0

//The software PWM is connected to PIN 12. You cannot use the pin 12 if you are using
//a Braccio shield V4 or newer
#define SOFT_START_CONTROL_PIN     12

//Low and High Limit Timeout for the Software PWM
#define LOW_LIMIT_TIMEOUT 2000
#define HIGH_LIMIT_TIMEOUT 6000

class _Braccio {

public:
  _Braccio();

  /**
  * Braccio initializations and set intial position
  * Modifing this function you can set up the initial position of all the
  * servo motors of Braccio
  *@param soft_start_level: the minimum value is -70, default value is 0 (SOFT_START_DEFAULT)
  * You should set begin(SOFT_START_DISABLED) if you are using the Arm Robot shield V1.6
  */
```

```
  unsigned int begin(int soft_start_level=SOFT_START_DEFAULT);

  /**
   * This function allow the user to control all the servo motors in the Braccio
   */
  int ServoMovement(int delay, int Vbase,int Vshoulder, int Velbow, int Vwrist_ver, int Vwrist_rot, int Vgripper);


private:
  /*
  * This function, used only with the Braccio Shield V4 and greater,
  * turn ON the Braccio softly and save Braccio from brokes.
  * The SOFT_START_CONTROL_PIN is used as a software PWM
  *  @param soft_start_level: the minimum value is -70, , default value is 0 (SOFT_START_DEFAULT)
  */
  void _softStart(int soft_start_level);

  /*
  * Software implementation of the PWM for the SOFT_START_CONTROL_PIN,HIGH
  * @param high_time: the time in the logic level high
  * @param low_time: the time in the logic level low
  */
  void _softwarePWM(int high_time, int low_time);


};

extern _Braccio Braccio;

#endif // BRACCIO_H_
```

- **Braccio.cpp**

```
#include "Braccio.h"

extern Servo base;
extern Servo shoulder;
extern Servo elbow;
extern Servo wrist_rot;
```

```
extern Servo wrist_ver;
extern Servo gripper;

extern int step_base = 0;
extern int step_shoulder = 45;
extern int step_elbow = 180;
extern int step_wrist_rot = 180;
extern int step_wrist_ver = 90;
extern int step_gripper = 10;


_Braccio Braccio;

//Initialize Braccio object
_Braccio::_Braccio() {
}

/**
 * Braccio initialization and set intial position
 * Modifing this function you can set up the initial position of all the
 * servo motors of Braccio
 * @param soft_start_level: default value is 0 (SOFT_START_DEFAULT)
 * You should set begin(SOFT_START_DISABLED) if you are using the Arm Robot shield
V1.6
 * SOFT_START_DISABLED disable the Braccio movements
 */
unsigned int _Braccio::begin(int soft_start_level) {
        //Calling Braccio.begin(SOFT_START_DISABLED) the Softstart is disabled and
you can use the pin 12
        if(soft_start_level!=SOFT_START_DISABLED){
                pinMode(SOFT_START_CONTROL_PIN,OUTPUT);
                digitalWrite(SOFT_START_CONTROL_PIN,LOW);
        }

        // initialization pin Servo motors
        base.attach(4);
        shoulder.attach(2);
        elbow.attach(9);
        wrist_rot.attach(6);
        wrist_ver.attach(5);
        gripper.attach(3);

        //For each step motor this set up the initial degree
        base.write(0);
        shoulder.write(40);
        elbow.write(180);
        wrist_ver.write(170);
        wrist_rot.write(0);
        gripper.write(73);
        //Previous step motor position
        step_base = 0;
        step_shoulder = 40;
```

```
        step_elbow = 180;
        step_wrist_ver = 170;
        step_wrist_rot = 0;
        step_gripper = 73;

        if(soft_start_level!=SOFT_START_DISABLED)
                _softStart(soft_start_level);
        return 1;
}

/*
Software implementation of the PWM for the SOFT_START_CONTROL_PIN,HIGH
@param high_time: the time in the logic level high
@param low_time: the time in the logic level low
*/
void _Braccio::_softwarePWM(int high_time, int low_time){
        digitalWrite(SOFT_START_CONTROL_PIN,HIGH);
        delayMicroseconds(high_time);
        digitalWrite(SOFT_START_CONTROL_PIN,LOW);
        delayMicroseconds(low_time);
}

/*
* This function, used only with the Braccio Shield V4 and greater,
* turn ON the Braccio softly and save it from brokes.
* The SOFT_START_CONTROL_PIN is used as a software PWM
* @param soft_start_level: the minimum value is -70, default value is 0
(SOFT_START_DEFAULT)
*/
void _Braccio::_softStart(int soft_start_level){
        long int tmp=millis();
        while(millis()-tmp < LOW_LIMIT_TIMEOUT)
                _softwarePWM(80+soft_start_level, 450 - soft_start_level);   //the sum
should be 530usec

        while(millis()-tmp < HIGH_LIMIT_TIMEOUT)
                _softwarePWM(75 + soft_start_level, 430 - soft_start_level); //the sum
should be 505usec

        digitalWrite(SOFT_START_CONTROL_PIN,HIGH);
}

/**
 * This functions allow you to control all the servo motors
 *
 * @param stepDelay The delay between each servo movement
 * @param vBase next base servo motor degree
 * @param vShoulder next shoulder servo motor degree
 * @param vElbow next elbow servo motor degree
 * @param vWrist_ver next wrist rotation servo motor degree
 * @param vWrist_rot next wrist vertical servo motor degree
 * @param vgripper next gripper servo motor degree
```

```
 */
int _Braccio::ServoMovement(int stepDelay, int vBase, int vShoulder, int vElbow,int
vWrist_ver, int vWrist_rot, int vgripper) {

        // Check values, to avoid dangerous positions for the Braccio
        if (stepDelay > 30) stepDelay = 30;
        if (stepDelay < 10) stepDelay = 10;
        if (vBase < 0) vBase=0;
        if (vBase > 180) vBase=180;
        if (vShoulder < 15) vShoulder=15;
        if (vShoulder > 165) vShoulder=165;
        if (vElbow < 0) vElbow=0;
        if (vElbow > 180) vElbow=180;
        if (vWrist_ver < 0) vWrist_ver=0;
        if (vWrist_ver > 180) vWrist_ver=180;
        if (vWrist_rot > 180) vWrist_rot=180;
        if (vWrist_rot < 0) vWrist_rot=0;
        if (vgripper < 10) vgripper = 10;
        if (vgripper > 73) vgripper = 73;

        int exit = 1;

        //Until the all motors are in the desired position
        while (exit)
        {
                //For each servo motor if next degree is not the same of the previuos than
do the movement
                if (vBase != step_base)
                {
                        base.write(step_base);
                        //One step ahead
                        if (vBase > step_base) {
                                step_base++;
                        }
                        //One step beyond
                        if (vBase < step_base) {
                                step_base--;
                        }
                }

                if (vShoulder != step_shoulder)
                {
                        shoulder.write(step_shoulder);
                        //One step ahead
                        if (vShoulder > step_shoulder) {
                                step_shoulder++;
                        }
                        //One step beyond
                        if (vShoulder < step_shoulder) {
                                step_shoulder--;
                        }
```

```
}

if (vElbow != step_elbow)
{
	elbow.write(step_elbow);
	//One step ahead
	if (vElbow > step_elbow) {
		step_elbow++;
	}
	//One step beyond
	if (vElbow < step_elbow) {
		step_elbow--;
	}

}

if (vWrist_ver != step_wrist_rot)
{
	wrist_rot.write(step_wrist_rot);
	//One step ahead
	if (vWrist_ver > step_wrist_rot) {
		step_wrist_rot++;
	}
	//One step beyond
	if (vWrist_ver < step_wrist_rot) {
		step_wrist_rot--;
	}

}

if (vWrist_rot != step_wrist_ver)
{
	wrist_ver.write(step_wrist_ver);
	//One step ahead
	if (vWrist_rot > step_wrist_ver) {
		step_wrist_ver++;
	}
	//One step beyond
	if (vWrist_rot < step_wrist_ver) {
		step_wrist_ver--;
	}
}

if (vgripper != step_gripper)
{
	gripper.write(step_gripper);
	if (vgripper > step_gripper) {
		step_gripper++;
	}
	//One step beyond
	if (vgripper < step_gripper) {
		step_gripper--;
```

```
                        }
                }

                //delay between each movement
                delay(stepDelay);

                //It checks if all the servo motors are in the desired position
                if ((vBase == step_base) && (vShoulder == step_shoulder)
                                && (vElbow == step_elbow) && (vWrist_ver ==
step_wrist_rot)
                                && (vWrist_rot == step_wrist_ver) && (vgripper ==
step_gripper)) {
                        step_base = vBase;
                        step_shoulder = vShoulder;
                        step_elbow = vElbow;
                        step_wrist_rot = vWrist_ver;
                        step_wrist_ver = vWrist_rot;
                        step_gripper = vgripper;
                        exit = 0;
                } else {
                        exit = 1;
                }
        }
}
```

- **Servo.h**

```
#ifndef Servo_h
#define Servo_h

#include <inttypes.h>

/*
 * Defines for 16 bit timers used with  Servo library
 *
 * If _useTimerX is defined then TimerX is a 16 bit timer on the current board
 * timer16_Sequence_t enumerates the sequence that the timers should be allocated
 * _Nbr_16timers indicates how many 16 bit timers are available.
 */

// Architecture specific include
#if defined(ARDUINO_ARCH_AVR)
#include "avr/ServoTimers.h"
#elif defined(ARDUINO_ARCH_SAM)
#include "sam/ServoTimers.h"
#elif defined(ARDUINO_ARCH_SAMD)
#include "samd/ServoTimers.h"
#elif defined(ARDUINO_ARCH_STM32F4)
```

```
#include "stm32f4/ServoTimers.h"
#elif defined(ARDUINO_ARCH_NRF52)
#include "nrf52/ServoTimers.h"
#else
#error "This library only supports boards with an AVR, SAM, SAMD, NRF52 or STM32F4 processor."
#endif

#define Servo_VERSION           2     // software version of this library

#define MIN_PULSE_WIDTH       544     // the shortest pulse sent to a servo
#define MAX_PULSE_WIDTH      2400     // the longest pulse sent to a servo
#define DEFAULT_PULSE_WIDTH  1500     // default pulse width when servo is attached
#define  REFRESH_INTERVAL        20000         // minumim time to refresh servos in microseconds

#define SERVOS_PER_TIMER       12     // the maximum number of servos controlled by one timer
#define MAX_SERVOS   (_Nbr_16timers  * SERVOS_PER_TIMER)

#define INVALID_SERVO         255     // flag indicating an invalid servo index

#if !defined(ARDUINO_ARCH_STM32F4)

typedef struct  {
  uint8_t nbr        :6 ;           // a pin number from 0 to 63
  uint8_t isActive   :1 ;           // true if this channel is enabled, pin not pulsed if false
} ServoPin_t   ;

typedef struct {
  ServoPin_t Pin;
  volatile unsigned int ticks;
} servo_t;

class Servo
{
public:
  Servo();
  uint8_t attach(int pin);          // attach the given pin to the next free channel, sets pinMode, returns channel number or 0 if failure
```

```
  uint8_t attach(int pin, int min, int max); // as above but also sets min and max values for writes.
  void detach();
  void write(int value);             // if value is < 200 its treated as an angle, otherwise as pulse width in microseconds
  void writeMicroseconds(int value); // Write pulse width in microseconds
  int read();                        // returns current pulse width as an angle between 0 and 180 degrees
  int readMicroseconds();            // returns current pulse width in microseconds for this servo (was read_us() in first release)
  bool attached();                   // return true if this servo is attached, otherwise false
private:
   uint8_t servoIndex;               // index into the channel data for this servo
   int8_t min;                       // minimum is this value times 4 added to MIN_PULSE_WIDTH
   int8_t max;                       // maximum is this value times 4 added to MAX_PULSE_WIDTH
};

#endif
#endif
```